ONLINE ENSEMBLE LEARNING IN THE PRESENCE OF CONCEPT DRIFT

by

LEANDRO LEI MINKU

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
Engineering and Physical Sciences
The University of Birmingham
July 2010

Abstract

In online learning, each training example is processed separately and then discarded. Environments that require online learning are often non-stationary and their underlying distributions may change over time (concept drift). Even though ensembles of learning machines have been used for handling concept drift, there has been no deep study of why they can be helpful for dealing with drifts and which of their features can contribute for that. The thesis mainly investigates how ensemble diversity affects accuracy in online learning in the presence of concept drift and how to use diversity in order to improve accuracy in changing environments. This is the first diversity study in the presence of concept drift. The main contributions of the thesis are:

- An analysis of negative correlation in online learning.
- A new concept drift categorisation to allow principled studies of drifts.
- A better understanding of when, how and why ensembles of learning machines can help to handle concept drift in online learning.
- Knowledge of how to use information learnt from the old concept to aid the learning of the new concept.
- A new approach called Diversity for Dealing with Drifts (DDD), which is accurate both in the presence and absence of drifts.

Acknowledgements

Firstly, many thanks to my supervisor, Xin Yao, whose advice on conducting research has been excellent. I would also like to thank him for his professionalism, generosity and understanding. He has shown to be a respectful person not only for his achievements, but also for his personality.

Many thanks also to the thesis group members (Peter Tiňo and John Bullinaria) for their helpful discussions. The current and previous members of the Data Mining Reading Group (Huanhuan Chen, Shuo Wang, Yang Yu, Naiqian Li, Jiali Yun and Rodrigo Soares), Allan White, Hirotaka Inoue and Ke Tang also provided very fruitful discussions. Siang Yew Chong helped me a lot as well, especially during the first year of my course. Thank you very much!

Thank you also to Nikunj Oza for making his online bagging algorithm available, and to Manuel Baena-García, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, Ricard Gavald, and Rafael Morales-Bueno for making their Early Drift Detection Method open source.

I would like to thank not only people who contributed with the technical aspects of my research, but also all the administrative staff of the School of Computer Science, in particular Joan Beard, Justine James and Sharon Powell, who have always been kind and made me feel welcomed. Thank you also to Steve Vickers for this role as Research Students Tutor and to Achim Jung for his role as Head of School.

Thanks very much also to my host during the internship at Google, Harmut Maennel, and to all my colleagues there. The experience that I gained during those six months was very worthwhile and everyone there was always willing to help.

I would also like to show my deepest gratitude to my family, who even being far away always tried to hear me and to understand the problems I faced. Thanks very much to Carmen Schenberger, Priscila Minku and Chiho Yoneyama.

Finally, I am also very grateful to my friends for the several conversations, both related to life in general and to life in academia. Siang Yew Chong showed me all the city and introduced me to a very important thing: the cafes! Thanks to him, I realized that a very nice mug of coffee is a great combination to research =) Yang Li has always been a very warm person, showing to have a big heart. Bryony O'Hara and Deborah Johnstone helped me not only with conversations,

but also with moving all my belongings from one accommodation to another. Many thanks to these and to all my friends who supported me during these last years.

This work was supported by the Overseas Research Students Award Scheme (ORSAS) and a School of Computer Science Scholarship.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ACE ..... Adaptive Classifiers-Ensemble System
AdaBoost ..... Adaptive Boosting
AddExp ..... Addictive Expert Ensembles
ANOVA ..... Analysis of Variance
ART ..... Adaptive Resonance Theory
CDC ..... Concept Drift Committee
DWM ..... Dynamic Weight Majority
EDDM ..... Early Drift Detection Method
EFuNN ..... Evolving Fuzzy Neural Network
ESGNN ..... Ensemble of Self-Generating Neural Networks
GA ..... Genetic Algorithm
IB3 ..... Instance-based Learning 3
ITI ..... Incremental Tree Inducer
LCS ..... Learning Classifier Systems
MadaBoost ..... Modified Adaptive Boosting
MLP ..... Multi-Layer Perceptron
MSE ..... Mean Squared Error
NB ..... Naive Bayes
NCL ..... Negative Correlation Learning
Pdf ..... Probability Density Function
SEA ..... Streaming Ensemble Algorithm
SGNN ..... Self-Generating Neural Network
SGNT ..... Self-Generating Neural Tree
SOM ..... Self-Organizing maps
SONG ..... Self-Organizing Neural Grove
STEPD ..... Statistical Test of Equal Proportions
Todi ..... Two On-line Classifiers for Learning and Detecting Concept Drift
UCS ..... Upervised Classier Systems

Chapter 1

Introduction

Machine Learning is an area within Artificial Intelligence which involves the study and development of computational models capable of improving their performance with experience and of acquiring knowledge on their own (Mitchell et al.; 1988). Examples of tasks which can be performed by machine learning approaches are prediction, classification, recognition, planning, etc.

Most machine learning approaches operate in offline mode. They first learn how to perform a particular task and then are used to perform this task. No task can be performed during the learning phase and, after the learning phase is completed, the system cannot further improve or change.

However, most practical problems change with time, i.e., they can suffer concept drift (Narasimhamurthy and Kuncheva; 2007). For example, consider an information filtering system which predicts the user's reading preferences. Users can change their preferences with time and a system which learnt how to predict them in the past will fail if it cannot update according to the new ones (Scholz and Klinkenberg; 2007a). Another example is a recommender or advertising system, in which customers' behaviour may change depending on the time of the year, on the inflation and on new products made available. Other examples include several other applications in which training data arrive continuously (streams of data), such as systems for computer security (Gao, Fan and Han; 2007), market-basket analysis (Angelov; 2006; Scholz and Klinkenberg; 2007a), spam detection (Nishida; 2008) and web pages classification (Ali et al.; 2006).

Differently from offline learning algorithms, online learning algorithms can be used to perform a particular task at the same time as the learning occurs. They are updated whenever a new training example is available, being able to perform lifelong learning in the sense that they do not ever need to stop learning. So, they can attempt to adapt to possible changes in the environment, if properly designed. Incremental learning algorithms can also operate in changing environments and are able to work in a more general scenario, in which training data can be processed in chunks. More detailed definitions are given in section 1.1.1.

From the above, we can observe that there is a vast number of applications that can benefit from online and incremental learning in changing environments. The thesis mainly concentrates in online learning, which can be used to solve both online and incremental problems, as explained in section 1.1.1. Other reasons for the choice of online learning are explained in chapter 2, and include the lack of stability of incremental learning approaches during stable concepts, delayed system update and drift handling, higher memory and time requirements. Due to its low time and memory requirements, besides being useful for applications in which training data arrive continuously (streams of data), online learning is also useful for applications with tight time and space restrictions, such as prediction of conditional branch outcomes in microprocessors (Fern and Givan; 2003).

Ensembles of learning machines are approaches which have been showing to outperform single learning machines both in offline (Liu and Yao; 1999a,b; Liu et al.; 1999; Chen and Yao; 2009; Breiman; 1996a; Schapire; 1990; Drucker et al.; 1992; Freund; 1995; Freund and Schapire; 1996b,a) and online mode (Blum; 1997; Lim and Harrison; 2003; Kotsiantis and Pintelas; 2004; Oza and Russell; 2001a,b, 2005; Fern and Givan; 2000, 2003; Lee and Clyde; 2004). They are sets of learners trained to perform the same task and combined with the aim of improving predictive performance (Chen and Yao; 2009). They have also been used specifically for dealing with changing environments (Stanley; 2003; Kolter and Maloof; 2003, 2007; Wang et al.; 2003; Scholz and Klinkenberg; 2005, 2007b; Ramamurthy and Bhatnagar; 2007; He and Chen; 2008).

The following points motivate us to use ensembles for online learning in changing environments:

- They have been successfully applied for offline learning. in a particular order, reducing the variance should particularly cause a good effect in this case.
- They can maintain several members with different performances considering a particular environmental state. Whereas some members may be less accurate for the current state, they may be more accurate for future states. If a system is carefully designed, it may be able to identify and/or use the members that become more accurate when the environment suffers a change in such a way to improve the system's performance.
- Online algorithms are frequently used for applications which require fast execution time, either because they have large amounts of data (Street and Kim; 2001) or because they have tight time restrictions (Fern and Givan; 2003). Ensembles naturally allow parallel implementation, which can be helpful for reducing their execution time. So, the benefit in accuracy that can be obtained by using several base learners does not necessarily mean that the time cost is much higher than a single learner's.

Even though ensembles have been widely studied and successfully used in offline mode, there is no deep study of why and when they can be useful for online learning in changing environments.

A better understanding of the behaviour of ensembles in online changing environments can reveal if their potential is being correctly used and allow better exploitation of their features. Such study is performed in the thesis and inspires a new and more accurate approach for dealing with concept drift in online learning.

This section is further divided as follows. Section 1.1 describes the problem of online learning in the presence of concept drift in more detail. Section 1.2 explains the research questions answered by the thesis. Section 1.3 explains the contributions and the organisation of the thesis.

1.1 Formulation of the Problem

Section 1.1.1 gives the definitions of incremental and online learning, section 1.1.2 gives the definition of concept drift, section 1.1.3 explains the notion of hidden contexts and section 1.1.4 explains the points in which the problem of concept drift needs better solutions.

1.1.1 Incremental and Online Learning Definition

The terms incremental and online learning have been used with (sometimes slightly) different meanings in the literature and are frequently mixed. According to Polikar et al. (2001), the term incremental learning has been used to indicate incremental network growing and pruning, online learning or relearning formerly misclassified examples. The terms constructive learning and lifelong learning have also been used to indicate incremental learning.

In this work, we will adopt the definition of incremental learning given by Polikar et al. (2001), as it is more clear than the general notions given by authors such as Littlestone (1988) and is very similar to the definitions given by Schaal and Atkeson (1998) and Kasabov (2003). According to Polikar et al. (2001), an incremental algorithm is an algorithm that meets the following criteria:

- It should be able to learn additional information from new data.
- It should not require access to the original data, used to train the existing classifier.
- It should preserve [useful] previously acquired knowledge (that is, it should not suffer from catastrophic forgetting).
- It should be able to accommodate new classes that may be introduced with new data.

Here, we perform a small modification to the definition, making clear that only *useful* previous knowledge should be preserved. It is particularly important to emphasize that in the context of changing environments, where the function being learnt can change over time.

Several definitions of online learning can also be found in the literature. Some examples are:

> Online learning is concerned with learning each data example separately as the system operates (usually in real-time), and the data may exist only for a short time. After observing each data example, the system makes changes in its structure ... to optimize the goal function...
>
> (Kasabov; 2003)

> ... *online* learning algorithms take as input a single labelled training instance as well as a hypothesis and output an updated hypothesis. Thus, given a sequence of training instances an online algorithm will produce a sequence of hypotheses.
>
> (Fern and Givan; 2003)

> Online learning algorithms process each training instance once "on arrival" without the need for storage and reprocessing, and maintain a current hypothesis that reflects all the training instances so far.
>
> (Oza and Russell; 2001a)

> Online prediction is a learning model in which an agent predicts the classification of a sequence of items and attempts to minimize the total number of prediction errors.
>
> (Freund and Schapire; 1996b)

> [Online l]earning proceeds in a sequence of trials. In each trial the algorithm receives an *instance* from some fixed *domain* and is to produce a binary *prediction*. At the end of the trial the algorithm receives a binary *label*, which can be viewed as the correct prediction for the instance.
>
> (Littlestone and Warmuth; 1994)

The terms continuous learning and real time learning have also been used to indicate online learning. We can observe that the online prediction definition given by Freund and Schapire (1996b) is very general and does not make restrictions about the way the learning is performed. We can also observe that Littlestone and Warmuth (1994) use a very strict definition, which could be considered as a particular scenario of online learning in which the same examples used for testing are also used for training. This scenario is also used in several other works, e.g., Kasabov (2003) and Fern and Givan (2003), although the definitions of online learning given in these works do not require the test set to be the same as the training set. We can also find many works which use a test set different from the training set, e.g., Oza and Russell (2001a) and Fern and Givan (2003).

In this work, we will adopt a combination of the definitions given by Oza and Russell (2001a) and Fern and Givan (2003): **Online learning algorithms process each training example once "on arrival" without the need for storage and reprocessing (Oza and Russell; 2001a). They take as input a single labelled training example as well as a hypothesis and output an updated hypothesis (Fern and Givan; 2003) of the function being learnt.** We will also consider that a time step corresponds to the presentation of one training example.

It is interesting to observe that, by using these definitions, online learning can be seen as a strict case of incremental learning. In the same way as incremental learning algorithms, online learning algorithms should be able to learn additional information from new data and to preserve useful previously acquired knowledge without requiring access to previous training examples. They should also be able to accommodate new classes. However, incremental learning algorithms can process new data in chunks of considerably large size, whereas online learning algorithms have to process each training example separately. This differentiation is particularly important because an incremental learning algorithm can process each chunk several times, possibly even using an offline learning algorithm for each chunk. So, the way to design online and incremental learning algorithms can be very different.

It is important not to confuse the term online learning used in this work with the term online learning related to the backpropagation algorithm, used to train Multi-Layer Perceptrons (MLPs) (Bishop; 2005). In the online backpropagation algorithm (Wilson and Martinez; 2003) (also called stochastic backpropagation (LeCun et al.; 1998)), the weights of an MLP are updated right after the presentation of each training example. In the batch (standard) backpropagation, the weights are updated only after the presentation of all training examples. However, both the online and batch versions of backpropagation can process the whole set of training examples several times (epochs). This behaviour cannot be considered as online learning according to our definition. In order not to confuse the online backpropagation algorithm with online learning, we will refer to it as stochastic backpropagation. It is also worth noting that the stochastic backpropagation can be considered as an online algorithm when only one epoch is used for training or if each training example is used to update the weights of the MLP a certain number of times "on arrival" and then discarded, as done by Oza and Russell (2005).

It is also worth mentioning the difference between online/offline environments and online/offline algorithms. In online environments, data examples either continuously arrive or there are very tight space and time restrictions. As previously explained, if one tries to use an offline algorithm to model an online environment, the model cannot be continuously updated and/or adapted to changes. Besides, offline algorithms usually have higher space and time requirements. On the other hand, in offline environments, no more data is acquired after an initial data set is obtained. So, the amount of data is usually relatively small and can be processed many times. This type of environment can be modelled by either offline or online algorithms, even though offline algorithms are likely to obtain higher accuracy for being specifically designed

for this type of problem.

1.1.2 Concept Drift Definition

We will consider concept drift in the context of classification problems in the thesis and we will work especially with binary classification. Similarly to the term *online learning*, the terms *concept* and *concept drift* are used with different meanings in the literature.

According to Narasimhamurthy and Kuncheva (2007), some authors consider the term *concept* as the whole distribution of the problem in a certain time step. The whole distribution of the problem, which is characterized by the joint distribution $p(\mathbf{x}, w)$, where $\mathbf{x}$ are the input attributes and w are the classes, can be represented by:

- unconditional probability density function (pdf) $p(\mathbf{x})$, and
- posterior probabilities $P(w_i|\mathbf{x}),\ i = 1, 2, ..., c.$

or

- prior probabilities of the classes $P(w_1), ..., P(w_c)$, where c is the number of existing classes, and
- class-conditional pdfs $p(\mathbf{x}|w_i),\ i = 1, ..., c$

By adopting this definition of concept, the term *concept drift* represents any changes in the distribution of the problem (Gama et al.; 2004). The term *population drift* (Kelly et al.; 1999) can also be used to represent changes in the underlying distributions.

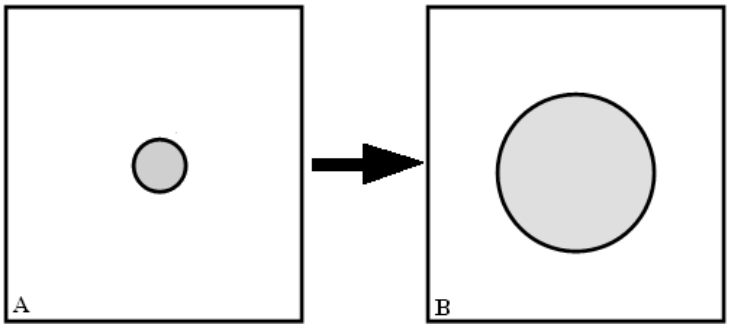
Some other authors consider that the term *concept* refers to the variables we want to predict, which in the context of classification problems are the target classes. So, a *concept drift* occurs when the class labels of the examples change over time (Kolter and Maloof; 2007). Although many papers do not provide explicit and clear definitions of concept and concept drift, we can consider that this definition was used by Schlimmer and Granger (1986) and various subsequent papers, such as Widmer and Kubat (1996), Kolter and Maloof (2003), (Stanley; 2003) and Tsymbal (2004).

In order to formalize this definition, we will use the joint distribution as well. Consider $p(\mathbf{x}, w_i) = p(\mathbf{x})P(w_i|\mathbf{x})$. A change in the target class w_i would be associated to a change in $P(w_i|\mathbf{x})$. However, a change in $p(\mathbf{x})$ would happen only if there are other environmental changes besides concept drift. Considering that $p(\mathbf{x})$ is fixed, a change in $P(w_i|\mathbf{x})$ is always associated to a change in $P(w_i)$ or $p(\mathbf{x}|w_i)$.

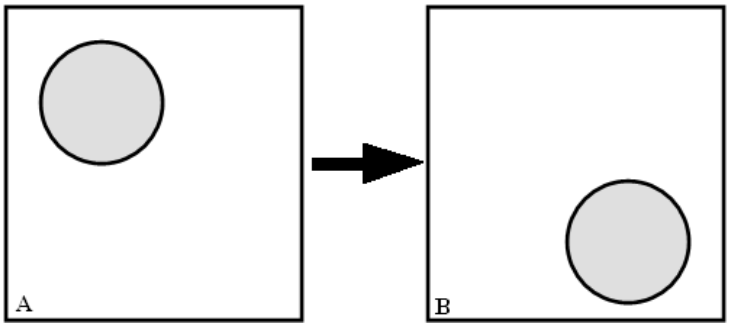
For example, consider a binary classification problem in which the training examples are of the form $[x, y, t]$, where x and y are input attributes representing a point in a unit square and t is a target class indicating whether the point (x, y) is inside or outside a circle represented by the equation $(x - a)^2 + (y - b)^2 \leq r$. We could have the two following drifts:

- Drift from concept A, in which $a = 0.5$, $b = 0.5$, $r = 0.1$, to concept B, in which $a = 0.5$, $b = 0.5$, $r = 0.3$. This drift is illustrated by the schematic figure 1.1(a).
- Drift from concept A, in which $a = 0.3$, $b = 0.3$, $r = 0.2$, to concept B, in which $a = 0.7$, $b = 0.7$, $r = 0.2$. This drift is illustrated by the schematic figure 1.1(b).

Both the drifts affect $P(w_i|\mathbf{x})$, where $w_i = \{0, 1\}$. However, the drift 1.1(a) affects both $P(w_i)$ and $p(\mathbf{x}|w_i)$, whereas the drift 1.1(b) affects $p(\mathbf{x}|w_i)$, but not in $P(w_i)$.



(a) Drift affecting $P(w_i)$ and $p(\mathbf{x}|w_i)$



(b) Drift affecting $p(\mathbf{x}|w_i)$

Figure 1.1: Examples of drift from concept A to concept B affecting $P(w_i|\mathbf{x})$, where $w_i = \{0, 1\}$. Grey color represents target class 1 and white color represents target class 0. In both cases, $p(\mathbf{x})$ remains fixed.

Kolter and Maloof (2007) mention that changes in the target class represent changes in the posterior probabilities $P(w_i|\mathbf{x})$ and in the class-conditional pdfs $p(\mathbf{x}|w_i)$. However, as we can see from the above examples, it should be made clear that a change in the target class can also be associated to a change in the prior probabilities of the classes $P(w_1), ..., P(w_c)$.

In the thesis, we will use the first definition of concept drift, which is more general, and we will refer to environments in which concept drift can occur as changing environments. The

reason for the choice is that a change in $p(\mathbf{x})$ when working with online learning can cause a change in the decision boundary learnt by an online algorithm. An online algorithm may learn a certain decision boundary because it received examples from a particular $p(\mathbf{x})$. However, a change in $p(\mathbf{x})$ may reveal that the current decision boundary is not good anymore, even if the target class of each example does not change. An online algorithm should be able to adapt its decision boundary when necessary.

1.1.3 The Notion of Hidden Contexts

Concept drifts may occur because the concept of interest depends on some *hidden context* (Widmer and Kubat; 1996), not given explicitly in the form of predictive features. For example, weather behaviour may vary depending on the season and image recognition may be influenced by illumination.

The context may suddenly or gradually become highly relevant and induce more or less radical changes in the target concepts. For example, a system which predicts accurately the weather during summer may give wrong predictions during winter and a system trained using images with similar illumination may become worse than random guess if images with different illumination need to be classified.

1.1.4 Desired Features of a Good Approach to Handle Concept Drift

Regarding changing environments, Widmer and Kubat (1996) affirm that:

> Effective learning in environments with hidden contexts and concept drift requires a learning algorithm that can detect context changes without being explicitly informed about them, can quickly recover from a context change and adjust its hypotheses to a new context, and can make use of previous experience in situations where old contexts and corresponding concepts reappear.

Nevertheless, the literature on concept drift shows that changes do not necessarily need to be detected explicitly. An example is the well known Streaming Ensemble Algorithm (SEA) (Street and Kim; 2001). Besides, learning machines have to be accurate not only when there is concept drift, but also when the concept is stable. Moreover, a system to deal with concept drift should be able to make use of previous experience not only when old concepts reappear, but whenever these experiences are useful for learning a certain concept. So, we can consider that online systems which try to handle concept drift should attempt to:

- Maximize accuracy when the concept is stable.

- Minimize the drop in accuracy when a concept drift occurs.
- Quickly recover from and adjust to concept drift.
- Efficiently use information previously learnt whenever it is beneficial.

Achieving all the above characteristics is a difficult task and there may be trade-offs. For example, a system which manages to attain very high accuracy while the environment is stable may have larger drop in the accuracy and take more time to recover from a drift. A system good for dealing with a certain type of drift may fail at dealing with another type of drift.

It is also worth noting that it is desirable for systems to continuously deal with drifts, so that they can recover and adjust quickly. For example, if a system is able to handle drifts only at every N time steps, it cannot recover quickly from drifts which begin before all the N examples have been received.

1.2 Research Questions

This section explains the research questions answered by this work and the motivation behind each of them. More detailed motivations for the way to tackle/solve each question/problem are given in the sections corresponding to the proposed solutions.

1.2.1 Negative Correlation in Incremental and Online Learning

The first research question answered by the thesis is: what are the strengths and weaknesses of negative correlation in incremental and online learning?

Negative Correlation Learning (NCL) (Liu and Yao; 1999b,a) is an ensemble learning approach which directly encourages diversity through the use of a penalty term in the error function of the neural networks belonging to the ensemble. It has shown to be able to outperform other ensemble methods in offline mode (Islam et al.; 2003; Wang et al.; 2004; Chandra and Yao; 2006). So, it is important to analyse whether we can also benefit from it in online or incremental environments. In particular, as NCL has a penalty term which allows encouraging more or less diversity, we should check whether this approach can be used for analysing the impact of diversity in online learning in the presence of concept drift, for answering research question 1.2.2.

Section 3.1 shows that NCL is promising in incremental learning. One of its strengths is that it can be used to overcome forgetting of knowledge previously learnt, which is an important problem related to incremental learning. However, overcoming forgetting so as to achieve higher generalization is not straightforward. In order to do so, it is necessary to join the advantages of

using the entire ensemble to learn new data at the same time as no further learning is performed by the current ensemble members. One of the possibilities is to clone the current ensemble and use the clone to learn new data, while maintaining the current ensemble with no further training. The two ensembles can be combined and pruned to a fixed size by using a selection procedure. In this way, it is possible to further improve accuracy at the same time as reducing forgetting.

Section 3.2 shows that NCL by itself is likely to be less helpful in online learning, having to be usually combined with other online learning approaches in order to provide good accuracy when the databases are not large. When combined with online bagging, NCL is a parallel-generation multiple-update approach which sends a different sequence of training examples for each base learner. In this way, higher accuracy than the direct application of NCL to online learning can be achieved through the use of more appropriate base learners. However, in this case, the NCL penalty term is not the only source of diversity. In order to study the impact of diversity in online learning in the presence of concept drift, it is desirable to use a considerably accurate approach which contains a parameter that allows consistently tuning diversity. So, NCL should not be used as the basis of such a study.

1.2.2 Ensembles for Dealing with Concept Drift

Another research question answered by the thesis is: when, how and why can ensembles be helpful for dealing with drifts?

Even though ensembles have been used to handle concept drift, the literature does not contain any deep study of why they can be helpful for that and which of their features can or cannot contribute for dealing with concept drift. Such a study is important because a better understanding of the behaviour of ensembles in the presence of concept drift allows better exploitation of their features for dealing with drifts.

One of the ensemble properties which may be helpful for dealing with drifts is diversity. A popular measure of diversity is Q statistic (Yule; 1900; Kuncheva and Whitaker; 2003). Considering two classifiers D_i and D_k, the Q statistic can be calculated as:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} , \tag{1.1}$$

where $N^{a,b}$ is the number of training examples for which the classification given by D_i is a and the classification given by D_k is b, 1 represents a correct classification and 0 represents a misclassification. Classifiers which tend to classify the same examples correctly will have positive values of Q, whereas classifiers which tend to classify different examples incorrectly will have negative values of Q. For an ensemble of L classifiers, the averaged Q statistic over all pairs of classifiers can be used as a measure of diversity. A higher average indicates less diversity and lower indicates more diversity.

In offline mode, diversity among base learners is an issue that has been receiving lots of attention in the ensemble learning literature. Many authors believe that the success of ensemble algorithms depends on both the accuracy and the diversity among the base learners (Dietterich; 1997; Kuncheva and Whitaker; 2003). However, no study of diversity has ever been done in online changing environments.

In the present thesis, a study of diversity in the presence of concept drift is presented. It aims at determining the differences between the role of diversity before and after a drift; whether diversity by itself can provide any advantage to handle concept drift considering the points mentioned in section 1.1.4; and the effect of diversity under different drift conditions.

Section 5.3 shows that it is possible to consistently encourage more or less diversity in online ensemble learning by using a modified version of the online bagging algorithm presented in section 2.4.1. Then, it shows that, before the drift, ensembles with less diversity obtain lower test errors. On the other hand, it is a good strategy to maintain highly diverse ensembles to obtain lower test errors shortly after the drift independent of the type of drift, even though high diversity tends to be more important for drifts which cause big changes. Longer after the drift, high diversity becomes less important. Diversity by itself can help to reduce the initial increase in error caused by a drift, but does not provide a fast recovery from drifts in the long term.

The diversity study presented in 5.3 shows the (different) behaviours of low and high diversity ensembles in relation to the old and new concepts by analysing the test error on the old or new concepts. Nevertheless, online learning often operates in the scenario explained by Littlestone and Warmuth (1994) and further adopted in many works, such as Kasabov (2003); Fern and Givan (2003); Gama et al. (2004); Baena-García et al. (2006):

> Learning proceeds in a sequence of trials. In each trial the algorithm receives an *instance* from some fixed *domain* and is to produce a binary *prediction*. At the end of the trial the algorithm receives a binary *label*, which can be viewed as the correct prediction for the instance.

Several real world applications operate in this sort of scenario, such as spam detection, prediction of conditional branches in microprocessors, information filtering, face recognition, etc. So, it is important to analyse the prequential accuracy, which considers the prediction done at each trial, as defined by equation 5.1 in section 5.4.1. Besides, if there are periods of time in which both the old and new concepts are active at the same time (gradual drifts), the system might be required to make predictions on instances belonging to both the old and new concepts, emphasizing the importance of analysing the prequential accuracy, which considers examples of both concepts at the same time when the drift is gradual. So, it is also important to check how a low and a high diversity ensemble would behave considering the prequential accuracy under different drift conditions. This study is performed in section 5.4, at the same time as answering research question 1.2.3.

1.2.3 Using the Old Concept to Aid the Learning of the New Concept

The following research question is also answered by the thesis: can we use information from the old concept to better deal with the new concept? How?

Intuitively, if a concept drift causes few changes to the old concept, information learnt from the old concept should be helpful to aid the learning of the new concept. However, simply training on the new concept a system that learnt well the old concept does not provide good convergence to the new concept, as shown by the study presented in section 5.3 and, indirectly, by the literature on concept drift (Kolter and Maloof; 2007; Baena-García et al.; 2006). To the best of our knowledge, no approach in the literature attempts to use information from the old concept in order to aid the learning of the new concept.

Section 5.4 shows that highly diverse ensembles trained on the old concept are able to converge to the new concept, as long as they start learning the new concept with low diversity. Considering the prequential accuracy, these ensembles are usually the best when (1) the drifts do not cause many changes or (2) long after drifts in which both the old and new concepts are kept active for a certain period of time. In these cases, they are even likely to be more accurate than new ensembles created from scratch when the drift begins, which is the strategy adopted by many approaches in the literature (Gama et al.; 2004; Baena-García et al.; 2006; Nishida and Yamauchi; 2007b). New low diversity ensembles created from scratch when the drift begins are usually the most accurate when the drift causes very big changes and occurs suddenly. Low diversity ensembles trained on the old concept are usually the most accurate shortly after the beginning of drifts which are not completed suddenly.

1.2.4 New Approach for Dealing with Drifts

The last research question answered by the thesis is: is it possible to create an approach more robust and accurate considering different types of drift and, at the same time, achieve good accuracy in the absence of drifts?

The answers to the research questions 1.2.2 and 1.2.3 provide knowledge about how diversity can be beneficial for several different types of drift. However, it is still necessary to join all this knowledge in order to create a new approach which is robust and accurate considering different kinds of drift, at the same time as it has good accuracy in the absence of drifts.

Section 6 proposes a new online ensemble learning approach to handle concept drifts called Diversity for Dealing with Drifts (DDD). This is the main contribution of the thesis. DDD aims at exploiting diversity to handle drifts, being more robust to false alarms (false positive drift detections) and having faster recovery from drifts. Experiments with artificial and real world data show that DDD always obtained similar or better accuracy than two other approaches from the concept drift literature both under several drift conditions and in the absence of drifts, with

very few exceptions.

1.3 Thesis Contributions and Organisation

In summary, the contributions of the thesis (and its organisation) are:

- An analysis of negative correlation in incremental and online learning, showing its strong and weak points (Minku et al.; 2009; Tang et al.; 2009; Minku and Yao; 2008). This is an answer to research question 1.2.1 and is presented in chapter 3.
- A new concept drift categorisation, separating drifts according to different criteria into mutually exclusive and non-heterogeneous categories (Minku et al.; 2010). The categorisation is presented in chapter 4, section 4.1.
- New artificial data sets which present different types of drift, allowing principled evaluation of approaches to handle concept drifts (Minku et al.; 2010). The data sets are presented in chapter 4, section 4.2.2.
- A technique to directly encourage more or less diversity in online bagging ensembles. The technique is presented in chapter 5, section 5.2, and allows us to consistently "tune" diversity.
- A better understanding of when, how and why ensembles of learning machines can help to handle concept drift in online learning, through a diversity study in the presence of concept drift (Minku et al.; 2010; Minku and Yao; 2010). This is an answer to research question 1.2.2 and is explained in chapter 5, sections 5.3 and 5.4.
- Knowledge about how to use information from the old concept in order to aid the learning of the new concept (Minku and Yao; 2009, 2010). This is an answer to research question 1.2.3 and is presented in chapter 5, section 5.4.
- A new approach to deal with concept drift using diversity (Minku and Yao; 2009, 2010). The approach is accurate and considerably robust against false positive drift detections in comparison to other approaches in the literature. This is an answer to research question 1.2.4 and is presented in chapter 6.

The main contributions are the answers to the research questions 1.2.1 to 1.2.4, in particular the study of ensemble learning in the presence of concept drift (chapters 5 and 6). The thesis also contains a chapter with literature review (chapter 2), and a chapter with conclusions and future work (chapter 7).

Some of the material presented in this thesis were published in the following papers:

1. **Minku, F. L., Inoue, H. and Yao, X. (2009).** Negative correlation in incremental learning, Natural Computing Journal - Special Issue on Nature-inspired Learning and Adaptive Systems 8(2): 289–320.

2. **Tang, K., Lin, M., Minku, F. L. and Yao, X. (2009).** Selective negative correlation learning approach to incremental learning, Neurocomputing 72: 2796–2805.

3. **Minku, F. L. and Yao, X. (2008).** On-line bagging negative correlation learning, Proceedings of the International Joint Conference on Neural Networks (IJCNN08), Part III, Hong Kong, pp. 1375–1382.

4. **Minku, L. L., White, A. and Yao, X. (2010).** The impact of diversity on on-line ensemble learning in the presence of concept drift, IEEE Transactions on Knowledge and Data Engineering (TKDE) 22: 730–742.

5. **Minku, F. L. and Yao, X. (2009).** Using diversity to handle concept drift in on-line learning, Proceedings of the International Joint Conference on Neural Networks (IJCNN09), Atlanta, pp. 2125–2132.

6. **Minku, L. L. and Yao, X. (2010).** DDD: A new ensemble approach for dealing with concept drift, IEEE Transactions on Knowledge and Data Engineering (TKDE), p. 16p. (accepted).

Chapter 2

Literature Review

Several approaches have been proposed to handle concept drift over the last years. Most of them are incremental learning approaches, which require an entire chunk of data to be available for learning, instead of processing each example separately. For instance, we can cite ensemble approaches which add new classifiers for new chunks of data and weight them according to their accuracy on recent data (Wang et al.; 2003; Scholz and Klinkenberg; 2005, 2007b; Ramamurthy and Bhatnagar; 2007; He and Chen; 2008), possibly using a boosting-like mechanism for the learning (Scholz and Klinkenberg; 2005, 2007b; He and Chen; 2008; Chu and Zaniolo; 2004).

Another example is Gao, Fan and Han (2007)'s work, which proposes the use of unweighted ensembles, as new data may belong to a concept different from the most recent training data. Street and Kim (2001) also report that no consistent improvement on the accuracy was obtained by their method when using weights.

Fan (2004)'s approach Stream Miner explicitly considers that one of the following four combinations of situations may occur when working with changing environments: new data is or is not sufficient by itself and there is or there is no concept drift. So, instead of always increasing the ensemble size, whenever a new chunk of data is available, Stream Miner explicitly compares the accuracy of a new classifier which learns only the new data chunk, the old classifier, the old classifier updated with the new chunk of data and a new classifier which learns both the new data and a subset of the old data. The problem of this strategy is that it makes it necessary to store an unlimited number of old examples for further training. This number can become very large, increasing the running time and memory requirements. Besides, as old examples are not kept in case they are not beneficial, there is no strategy to benefit from recurrent concepts.

Some approaches, such as Forman (2006) and Ramamurthy and Bhatnagar (2007), consider the existence of recurrent concepts more explicitly. Forman (2006) creates a new classifier to learn each new chunk of data and augments the set of features of the new chunk with P features, generated by the predictions that the P previous classifiers would have made. In this way, knowledge of previous base learners can be used for the training of the new base learner.

Ramamurthy and Bhatnagar (2007), on the other hand, explicitly maintain a global set of classifiers which can be used if they are accurate enough on the current chunk of data. New classifiers are included only if neither the ensemble formed by the classifiers in the global set nor the classifiers by themselves are considered accurate enough.

Some approaches also try to handle skewed distributions, which are frequently present in data stream learning (Gao, Fan, Han and Yu; 2007).

However, determining the chunk size in the presence of concept drifts is not a straightforward task. A too small chunk does not provide enough data for a new classifier to be accurate, whereas a too large chunk may contain data belonging to different concepts, making the adaptation to new concepts slow. Even though Scholz and Klinkenberg (2005, 2007b)'s approach allows the most recent classifier to learn the new chunk of data depending on a decision rule, instead of always creating a new classifier to learn a new chunk of data, the authors also reported that the approach does not perform well when the data chunks are too small. So, this approach also has the problem of determining the chunk size. Besides, no drift can be detected before a whole new chunk of data is received, i.e., these approaches cannot deal with drifts continuously, as explained to be important for having quick recovery from drifts in section 1.1.4.

Another problem is that most incremental learning approaches give little attention to the stability of the classifiers, giving more emphasis to the plasticity when they allow only a new classifier to learn a new chunk of data. While this could be desirable when drifts are very frequent, it is not a good strategy when drifts are less frequent than the chunk size, as at most one ensemble member learns each new training example. Offline methods such as bagging and boosting have the property that a single training example can contribute to the training of many ensemble members. Such a property is believed to be essential for obtaining a fast convergence to the desired target concept (Fern and Givan; 2003). In order for incremental learning approaches to be able to achieve similar accuracy to approaches which allow several ensemble members to learn the available data during stable concepts, each ensemble member would have to be trained with a large amount of data. However, large chunks of data are not good when trying to deal with drifts, as explained in the previous paragraphs. Section 3.1 illustrates this problem.

Even though there are a few incremental approaches which are more careful with the stability of the system when the concept is stable, such as Scholz and Klinkenberg (2007b)'s, online learning approaches tend to give more importance to stability when the concept is stable, by allowing the whole system to learn all the new incoming data when it is believed that there is no concept drift (Kolter and Maloof; 2003, 2005; Baena-García et al.; 2006; Gama et al.; 2004; Nishida and Yamauchi; 2007b).

Besides the need to determine the chunk size, the non continuity in handling drifts and the lack of strategy to deal with periods of stability, incremental learning approaches also suffer from delayed system update (it is necessary to wait for a whole new chunk of data in order to update the system) and higher memory and time requirements (the new chunks of data have to

be stored in memory and are usually processed many times).

As there are some applications which present inherent incremental environments and allow heavier memory and time requirements, incremental learning is still an important area of study. However, besides the fact that incremental learning approaches present the problems explained above, online approaches can be used to solve both online and incremental problems. So, the thesis concentrates on online learning.

The rest of this chapter is organized as follows: section 2.1 explains previous work on diversity in offline learning. Such work is related to the thesis because a novel study of diversity in changeable environments is presented. Section 2.2 presents Negative Correlation Learning (NCL), which is an ensemble learning approach whose behaviour when applied to incremental and online learning is investigated in the thesis. Section 2.3 presents incremental approaches used for comparisons in the study of NCL in incremental learning. Section 2.4 presents online ensemble learning approaches which do not attempt to handle concept drifts, in particular online bagging approaches which are closely related to the work presented in the thesis. Section 2.5 discusses online approaches which attempt to handle concept drifts. Section 2.6 presents a summary of the literature review.

2.1 Diversity in Offline Ensemble Learning

In offline mode, diversity among base learners is an issue that has been receiving lots of attention in the ensemble learning literature. The success of ensemble learning algorithms is believed to depend both on the accuracy and on the diversity among the base learners (Dietterich; 1997) and some empirical studies revealed that there is a positive correlation between accuracy of the ensemble and diversity among its members (Dietterich; 2000; Kuncheva and Whitaker; 2003). Breiman (2001) also shows that random forests with lower generalization error have lower correlation among base learners and higher base learners' strength. Besides, he derives an upper bound for the generalization error of random forests which depends on both correlation and strength of the base learners.

In regression tasks, the bias-variance-covariance decomposition (Ueda and Nakano; 1996) can provide a solid quantification of diversity for linearly weighted ensembles. The decomposition shows that the mean squared error of an ensemble depends critically on the amount of correlation between networks, quantified in the covariance term and that, ideally, we would like to decrease the covariance at the same time as being careful not to increase the bias and the variance terms.

However, there is still no clear analogue of the bias-variance-covariance decomposition when the predictors output discrete labels, as the concept of covariance is not defined. Although there are some theoretical results, they are highly restricted and make strong assumptions unlikely to hold in practice (Brown, Wyatt, Harris and Yao; 2005). It is still not clear how to define a diver-

sity measure for classifiers and how to use it to improve the ensemble's accuracy. Kuncheva and Whitaker (2003) analyse ten different diversity measures for classifiers. The authors empirically show that all these measures are correlated to each other, although some of them can exhibit a different behaviour from the others. An example of a popular diversity measure is given by equation 1.1, in section 1.2.2.

Tang et al. (2006) study the relationship between diversity and margins of an ensemble considering six different diversity measures for classifiers. The authors show that, when the average classification accuracy of the ensemble members on the training data is considered a constant and the maximum diversity is achievable, maximizing the diversity is equivalent to maximizing the minimum margin of the ensemble on the training examples. So, seeking diversity might be viewed as an implicit way to maximize the minimum margin of the ensemble. However, they theoretically and empirically show that the maximum diversity is usually not achievable. Besides, the minimum margin of an ensemble does not monotonically increase with respect to diversity. Hence, enlarging diversity is not exactly the same as enlarging the minimum margin. Based on that, they conclude that large diversity may not always correspond to better generalization performance.

Furthermore, it is usually affirmed in the literature that there is a trade-off between base learners' accuracy and diversity, meaning that lower accuracy may correspond to higher diversity. However, it is shown by Tang et al. (2006) that the relationship between accuracy and diversity is not straightforward and lower classification accuracy of ensemble members may not correspond to a higher diversity.

As we can see, the relationship among diversity, accuracy on the training set and generalization is complex. Nevertheless, it is important to study the effect of diversity not only in offline, but also in online changing environments, as the effect of diversity can be very different in these two cases. The literature does not contain such study.

2.2 Negative Correlation Learning (NCL)

Negative Correlation Learning (NCL) (Liu and Yao; 1999b,a) is an ensemble learning approach which directly encourages diversity through the use of a penalty term in the error function of the neural networks belonging to the ensemble. As NCL is able to outperform single neural networks and other ensemble methods in offline mode (Liu and Yao; 1999b,a; Islam et al.; 2003; Wang et al.; 2004), it is important to analyse whether we can benefit from it also in online or incremental environments. Such a study is presented in chapter 3.

NCL works as follows (Chandra et al.; 2006; Brown; 2004). Given a training set T of size N:

$$T = \{(x(1), d(1)), (x(2), d(2)), ..., (x(N), d(N))\} \ ,$$

where x ($x \in \mathbb{R}^p$) is the input to a neural network, d is the desired output and is a scalar. The assumption of d being a scalar is made to simplify the exposition of the ideas without loss of generality. Consider estimating d by forming an ensemble whose output is a simple average of a set of M neural network outputs[1]:

$$F(n) = \frac{1}{M} \sum_{i=1}^{M} F_i(n) \ , \tag{2.1}$$

where $F_i(n)$ is the output of the ith neural network and $F(n)$ is the output of the ensemble on the nth training example.

The aim of NCL is to produce a diverse ensemble, by inserting a penalty term which encourages diversity into the error function of each individual neural network. All neural networks are trained simultaneously and interactively on the same training set T. The error function E_i for the ith neural network in NCL is defined by the following equation:

$$E_i = \frac{1}{N} \sum_{n=1}^{N} E_i(n) = \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{2} (F_i(n) - d(n))^2 + \gamma p_i(n) \right) \ , \tag{2.2}$$

where $E_i(n)$ is the error of the ith neural network after the presentation of the nth training example. The first term in the right side of equation 2.2 is the empirical risk function of the ith neural network. The second term p_i is the correlation penalty function. The purpose of minimizing p_i is to penalize positive correlation of errors from different neural networks, i.e., to encourage negative correlation of a neural network error with the error of the rest of the ensemble. The parameter γ is used to adjust the strength of the penalty and it is problem-dependent (Brown, Wyatt and Tiño; 2005). The penalty function p_i may use the following equation:

$$p_i(n) = (F_i(n) - F(n)) \sum_{i \neq j} (F_j(n) - F(n)) \ . \tag{2.3}$$

Considering the property that the sum of deviations around a mean is zero, equation 2.3 can be rearranged to:

$$p_i(n) = -(F_i(n) - F(n))^2 \ . \tag{2.4}$$

[1]During negative correlation learning, a simple average is used to combine the neural network outputs. However, the combination method used by the ensemble during the test phase can be another one, e.g., majority vote.

The partial derivative of $E_i(n)$ with respect to the output of the network i on the nth training example is:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F_i(n) - d(n) - \gamma \left[2 \left(1 - \frac{1}{M} \right) (F_i(n) - F(n)) \right] \ . \tag{2.5}$$

When M is large, $(1 - 1/M)$ equals to 1. The standard Back-propagation algorithm (Rumelhart et al.; 1986) can be used with equation 2.5 for weight adjustments of the neural networks (Liu and Yao; 1999b,a), which can be, for example, Multi-Layer Perceptrons (MLPs). The weight updates of all neural networks are performed simultaneously. NCL has also been used in combination with constructive (Islam et al.; 2003) and evolutionary (Liu et al.; 1999) algorithms. Some other approaches were inspired on NCL in order to create ensembles using multi-objective evolutionary algorithms (Chandra and Yao; 2006).

In some papers (Liu and Yao; 1999b) previous to Brown (2004), the partial derivative was calculated as:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = F_i(n) - d(n) - \lambda(F_i(n) - F(n)) \ . \tag{2.6}$$

However, this calculation considers that $F(n)$ is constant with respect to $F_i(n)$, when it is actually not. When considering equation 2.6, it was believed that the strength parameter, which was called λ, was entirely problem-dependent. In order to calculate the partial derivative correctly, the strength parameter γ was introduced by Brown, Wyatt and Tiño (2005). Its relation to λ is shown in equation 2.7. As we can observe, λ is not entirely problem-dependent, as it has a deterministic component: $2(1 - \frac{1}{M})$. The parameter γ is still problem-dependent.

$$\lambda = \gamma \left[2 \left(1 - \frac{1}{M} \right) \right] \ . \tag{2.7}$$

Brown, Wyatt and Tiño (2005) mathematically showed that γ has an upper bound (equation 2.8):

$$\gamma_{upper} = \frac{M^2}{2(M-1)^2} \ . \tag{2.8}$$

The upper bound is used to avoid losing useful gradient information when negative correlation is used. The negative correlation penalty term 'warps' the error landscape of the neural network, making the global optimum hopefully easier to locate. However, if the landscape is warped too much, it could lose any useful gradient information. This state is indicated by the positive-definiteness of the Hessian matrix. The Hessian matrix describes the local curvature of a function of many variables. If it is positive definite at a certain point x, then the function has a local

minimum at x. Otherwise, it either has a local maximum at x or x is a point of inflexion. Assuming an estimator that is a linear combination of a number of nonlinear functions (e.g., a multi-layer perceptron with linear output nodes), Brown, Wyatt and Tiño (2005) showed that when $\gamma \geq \gamma_{upper}$, the Hessian matrix of the error function is never positive-definite, representing a loss of any useful gradient information.

The following observations can be made from equations 2.2, 2.3 and 2.5:

- During the training process, all individual neural networks interact with each other through their penalty terms in the error functions. Each neural network minimizes not only the difference between $F_i(n)$ and $d(n)$, but also the difference between $F(n)$ and $d(n)$, considering the error of all other neural networks while training a particular neural network.
- For $\gamma = 0$, the individual neural networks are trained independently.
- The ambiguity decomposition of the ensemble error can be used to explain why NCL works (Brown; 2004). Considering uniformly weighted ensembles, such as NCL, we have the following decomposition of the mean squared error (MSE) (Krogh and Vedelsby; 1995):

$$(F(n) - d(n))^2 = \sum_{i=1}^{M} \frac{1}{M}(F_i(n) - d(n))^2 - \sum_{i=1}^{M} \frac{1}{M}(F_i(n) - F(n))^2, \tag{2.9}$$

  where the first term represents the error of the ensemble members and the second term is the ambiguity of the ensemble members. If we multiply this equation by 1/2 and slightly rearrange it, we get:

$$\frac{1}{2}(F(n) - d(n))^2 = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{2}(F_i(n) - d(n))^2 - \frac{1}{2}(F_i(n) - F(n))^2\right). \tag{2.10}$$

  From this equation, we can see that the MSE of an ensemble can be decomposed into a summation where the i^{th} term is the individual ensemble member error plus the NCL penalty term (equation 2.4) using a strength parameter of $\gamma = 0.5$. So, minimizing the penalty term contributes to minimising the ensemble error.

- For $\gamma = 1$, we get from equation 2.5:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = \left(2 - \frac{2}{M}\right)F(n) + \left(-1 + \frac{2}{M}\right)F_i(n) - d(n) \ . \tag{2.11}$$

  The training of the neural networks minimizes the difference between $F(n)$ and $d(n)$ as well as the difference between $F_i(n)$ and $d(n)$. However, the minimization of $Fi(n) - d(n)$ is the cause for the minimization of $F(n) - d(n)$, as $F(n)$ is obtained from equation 2.1. So, it is possible to consider that $F_i(n)$ tends to $d(n)$. In this way, from equation 2.11, we get:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = \left(2 - \frac{2}{M}\right) F(n) + \left(-2 + \frac{2}{M}\right) d(n) = \left(2 - \frac{2}{M}\right) (F(n) - d(n)) \ . \tag{2.12}$$

When M is large, we get:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = 2(F(n) - d(n)) \ . \tag{2.13}$$

Note that the empirical risk function of the ensemble for the nth training example is:

$$E_{ens}(n) = \frac{1}{2} \left(\frac{1}{M} \sum_{i=1}^{M} F_i(n) - d(n) \right)^2 \ , \tag{2.14}$$

where M indicates the number of neural networks in the ensemble.

The partial derivative of $E_{ens}(n)$ with respect to $F_i(n)$ on the nth training example is:

$$\frac{\partial E_{ens}(n)}{\partial F_i(n)} = \frac{1}{M} \left(\frac{1}{M} \sum_{j=1}^{M} F_j(n) - d(n) \right) = \frac{1}{M} (F(n) - d(n)) \ . \tag{2.15}$$

In this case, we get:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = (2M - 2) \frac{\partial E_{ens}(n)}{\partial F_i(n)} \ . \tag{2.16}$$

And, when M is large:

$$\frac{\partial E_i(n)}{\partial F_i(n)} = 2M \frac{\partial E_{ens}(n)}{\partial F_i(n)} \ . \tag{2.17}$$

In other words, the minimization of the empirical risk function of the ensemble can be achieved by minimizing the error functions of individual neural networks. In effect, a large and more complex task of training the ensemble is automatically decomposed into a number of simpler tasks of training individual neural networks.

Chen and Yao (2009) observed that, even though the correlation term in the penalty function may seem to act as a regularization term, NCL does not totally overcome the problem of overfitting by tuning γ. So, they propose to add a regularization term to the ensemble error function in order to explicitly encourage decay of the weights of the NCL neural networks. The approach is theoretically founded on Bayesian inference and experiments show that it improves the NCL performance especially when the noise level is nontrivial in the data set.

2.3 Incremental Ensemble Learning Approaches

This section briefly describes three incremental learning approaches to which NCL is compared without considering concept drift in section 3.1: Learn++ (section 2.3.1), Self-Organizing Neural Grove (SONG) (section 2.3.2) and an evolutionary approach (section 2.3.3). Besides presenting the problems common to all incremental learning approaches as explained in the beginning of section 2, each specific approach has its particular successful points and problems, which are explained in the respective sections.

2.3.1 Learn++

A notable incremental ensemble learning algorithm is Learn++ (Polikar et al.; 2001). It creates multiple classifiers to each data chunk presented to the system. Inspired by AdaBoost (Freund and Schapire; 1996a, 1997), for each chunk, the training set for each base learner is created by sampling examples according to a distribution of probability.

However, in AdaBoost, the distribution of probability is built to give higher priority to instances misclassified by the last previously created classifier. In Learn++, the distribution is created considering the misclassification by the composite hypothesis, formed by all the classifiers created so far to that data chunk. In this way, incremental learning is possible, particularly when instances from new classes are introduced.

Successful points

Experiments on four benchmark databases showed that this approach was able to considerably improve the generalization from the first to the last data chunk without requiring access to previous data chunks. An additional analysis with one of the databases showed that Learn++ was able to attain better generalization than Fuzzy ARTMAP (Carpenter et al.; 1992).

Problems

One of the problems of Learn++ is that it uses a sequential generation of classifiers, both considering the classifiers generated to a particular data chunk and the ensembles generated to different data chunks. So, old ensemble members are not trained with new data and may have reduced accuracy, affecting the ensemble accuracy as a whole.

Another problem is that a new set of classifiers is created for each new data chunk. So, the ensemble size can become extremely large considering lifelong learning.

This approach does not allow tuning diversity. However, the different capacities of ensemble members to learn information from new data is one of the ensemble's features that motivates

its use for incremental and online learning. The level of diversity necessary to achieve fast and accurate learning in these environments may be different from the offline learning case and is likely to change as new data arrive. So, it is important not to simply attempt to mimic a diversity level that was successful in offline mode.

2.3.2 Self-Organizing Neural Grove (SONG)

Self-Organizing Neural Grove (SONG) (Inoue and Narihisa; 2003) is an ensemble of Self-Generating Neural Networks (SGNNs) (Wen et al.; 1992) which uses a pruning method for the structure of the SGNNs. In this way, it reduces the computation time and the memory requirements of Ensembles of Self-Generating Neural Networks (ESGNNs) (Inoue and Narihisa; 2000). SGNNs have simple network design and high learning speed. They are an extension of the self-organizing maps (SOM) of Kohonen (Kohonen; 1995) and use competitive learning, implemented as a Self-Generating Neural Tree (SGNT).

The SGNT algorithm is defined as the problem of how to construct a tree structure from the given data which consist of multiple attributes under the condition that the final leaves correspond to the given data. An ESGNN is constructed by presenting the training set using a different order to each of the SGNTs. The output of the ensemble is the majority vote of the outputs of the SGNTs. After the construction of the ensemble, a pruning method is used to compose a SONG. The one-pass learning of the SGNTs make it possible to directly apply SONG for incremental learning.

Successful points

SONG joins the advantages of ensembles to the advantage of using a base classifier that is adequate for incremental learning due to its one-pass learning. SONG trains all its members with all the data chunks, taking advantage from the differences among the members. The differences can be beneficial because some members can adapt better than the others depending on the incoming data.

The direct application of SONG to incremental learning has shown to be successful (Inoue and Narihisa; 2005). Experiments on a benchmark database showed that SONG improved the generalization from the first to the last data chunk and achieved better generalization than a single SGNN.

Problems

The ensemble diversity is based only on the order of presentation of the training examples. There is no evidence that this produces enough diversity. Besides, it is not possible to encourage more or less diversity, which is a desirable feature for incremental learning, as explained in section

2.3.1.

2.3.3 Evolutionary Approach

Seipone and Bullinaria (2005) developed an approach to incremental learning using evolving neural networks. An evolutionary algorithm is used to evolve some MLP parameters, such as the learning rates, initial weight distributions and error tolerance. The evolutionary process aims at evolving the parameters to produce networks with better incremental abilities. In each generation, the neural networks with the parameters codified by the evolutionary algorithms are trained using first the training examples of a particular chunk of data, then using another one and so on. While a neural network is being trained with a particular data chunk, the other chunks are not used in the training. However, each generation uses all the available chunks. The error produced by testing the neural networks with a validation set is used as a fitness measure.

Successful points

The approach is able to cope with new classes of data. Besides, it has shown to outperform Learn++ and a traditional MLP on a benchmark data set from the UCI Machine Learning Repository (Optical Digits) (Newman et al.; 2010).

Problems

Although each neural network is trained with only one data chunk at time, the evolutionary algorithm considers all the chunks in each generation of the evolution. So, even though the networks satisfy all the necessary criteria to incremental learning, the evolutionary algorithm does not.

2.4 Online Ensemble Approaches for Stable Concept Learning

There are several ensemble learning approaches which are proposed to deal with online environments, but do not have mechanisms specifically designed to handle concept drift. According to Fern and Givan (2003), online ensemble algorithms can be classified as sequential/parallel-generation approaches and single/multiple-update approaches. An algorithm takes a sequential-generation approach "if it generates the ensemble members one at a time, ceasing to update each member once the next one is started". Otherwise, it takes a parallel-generation approach. An online ensemble algorithm takes a single-update approach "if it updates only one ensemble member for each training instance encountered". Otherwise, it takes a multiple-update approach.

As Fern and Givan (2003) point out, sequential-generation and single-update approaches

present some problems. When the approach is sequential-generation, it is necessary to design a method to determine when each ensemble member should stop being trained, which is a difficult task. When the approach is single-update, it is likely to have slower convergence to the target concept, as only one of the ensemble members is adjusted at each time step. Offline methods such as bagging and boosting have the property that a single training example can contribute to the training of many ensemble members. Such a property is believed to be essential for obtaining a fast convergence to the desired target concept (Fern and Givan; 2003). So, in fact, approaches that are both sequential-generation and single update share some of the problems presented by many incremental learning approaches.

Some approaches originally developed to offline learning might also be claimed for use in online learning, e.g., Boosting by Filtering (Schapire; 1990) and Adaptive Boosting (AdaBoost) (Freund and Schapire; 1996a, 1997). In order to do so, a certain number of examples could be selected to train a first classifier. After that, a second classifier would be created by filtering new examples based on the previous classifier. The same process could be applied to create more classifiers. However, these approaches would be sequential-generation single-update approaches and would discard large amounts of data in the process of drawing the desired distribution of data to each ensemble member. So, they would need a long time to get new training examples that can be used for learning.

Modified Adaptive Boosting (MadaBoost) (Domingo and Watanabe; 2000) is a a modification of AdaBoost that can be used in the filtering framework without having extremely high execution time. This is achieved by bounding the weight that is attributed to the training examples. In this way, the time necessary for the filter to choose an example to be used for learning is reduced. However, MadaBoost is still a sequential-generation single-update approach. Other examples of sequential-generation single-update approaches are Pasting Small Votes (Breiman; 1999) and Streaming Random Forests (Abdulsalam et al.; 2007).

The problems presented by sequential-generation and single-update approaches are even more serious in the presence of concept drifts if no additional strategy is adopted to handle them. An ensemble which maintains immutable members (sequential-generation) would have difficulties to adapt to new concepts. Similarly, an ensemble in which only one of the members can be trained with each new training example (single-update) would have slow recovery from drifts. Nevertheless, even parallel-generation multiple-update approaches present slow adaptation to new concepts if no specific strategy to deal with concept drift is adopted, as shown in the concept drift literature (e.g., Kolter and Maloof (2003); Gama et al. (2004)) and in section 5.3.

In the thesis, we further divide parallel-generation multiple-update approaches into approaches which send the same sequence of training examples to each ensemble member and approaches which send a different sequence. Some of the approaches which send the same sequence consist of different ways of assigning weights for each ensemble member in an online way, e.g., Winnow (Littlestone; 1988) and Weight Majority (Littlestone and Warmuth; 1994). Others

are concerned with finding alternatives to create ensembles with different members when the same sequence of training examples is sent, e.g., Lim and Harrison (2003)'s and Kotsiantis and Pintelas (2004)'s work. Even though these approaches are parallel-generation multiple-update approaches, they have the problem that model diversity has to be built *a priori* rather than emerging from data itself.

The approaches which send a different sequence of training examples to each ensemble member are basically concerned with how to determine this sequence in an online way, so that model diversity does not need to be built *a priori*. Many of these approaches are online versions of well established offline approaches. As examples, we can cite online versions of boosting (Oza and Russell; 2001a,b, 2005; Fern and Givan; 2000, 2003) and bagging (Oza and Russell; 2001a,b, 2005; Fern and Givan; 2000, 2003; Lee and Clyde; 2004).

Among these approaches, online bagging was chosen to be used in the online diversity study presented in chapter 5. As further explained in that chapter, one of the reasons for opting to use this approach, instead of online versions of boosting, is that a simple modification including an additional parameter allows us to tune diversity. Besides, this parameter is the only source of diversity apart from the data set itself. So, it is possible to consistently increase or reduce diversity by varying it.

Oza and Russell (2001a,b, 2005)'s, Fern and Givan (2000, 2003)'s and Lee and Clyde (2004)'s online bagging algorithms are very similar to each other and are explained in section 2.4.1. Oza and Russell (2001a,b, 2005)'s and Fern and Givan (2000, 2003)'s online boosting algorithms are also similar to each other and are explained in section 2.4.2.

2.4.1 Online Bagging

Oza and Russell (2001a,b, 2005)'s online bagging is based on the fact that, when the number of training examples tends to infinity in offline bagging (Breiman; 1996a), each ensemble member h_m contains W copies of each original training example d, where the distribution of W tends to a $Poisson(1)$ distribution. Offline bagging's sampling distribution is $Multinomial(N, 1/N)$, for N draws. Online bagging's sampling distribution is $Poisson(N)$, for N draws. Oza and Russell (2001b) proves that, as the number of examples N tends to infinity, the probability-generating function of offline bagging's sampling distribution converges to the same function as the probability-generating function of online bagging's sampling distribution. That means that they have the same expectation when N tends to infinity.

For that reason, in online bagging, whenever a training example is available, for each ensemble member, a weight W is drawn from a $Poisson(1)$ distribution and the example is learnt in association to this weight. As $Poisson(1)$ provides discrete values, instead of presenting each training example once with a weight W to the base learning algorithm, it can be presented W times, so that the base learner does not need to deal with weights (algorithm 2.1). The

classification is done by unweighted majority vote, as in offline bagging.

Algorithm 2.1 Oza and Russell (2001a,b, 2005)'s Online Bagging

Inputs: ensemble **h**, ensemble size M, training example d and online learning algorithm for the ensemble members $OnlineBaseLearningAlg$.

1: **for** $m \leftarrow 1$ to M **do**
2: $W \leftarrow Poisson(1)$
3: **for** $w \leftarrow 1$ to W **do**
4: $h_m \leftarrow OnlineBaseLearningAlg(h_m, d)$
5: **end for**
6: **end for**

Output: updated ensemble **h**.

Fern and Givan (2000, 2003) proposed a very similar online bagging algorithm. The only difference is that it allows the choice between either using a Poisson(1) distribution or associating each example to $W = 1$ with probability P_u and $W = 0$ with probability $(1 - P_u)$, where P_u is chosen by the user.

Lee and Clyde (2004) proposed a lossless[2] online bayesian bagging algorithm. The algorithm works in a similar way to Oza and Russell (2001a,b, 2005)'s and Fern and Givan (2000, 2003) 's, but it draws weights from a $Gamma(1, 1)$ distribution. As the weights provided by $Gamma(1, 1)$ are continuous-valued on $(0, 1)$, the base learning algorithms need to be able to cope with weighted training examples, instead of receiving a certain number of repetitions of training examples.

2.4.2 Online Boosting

Oza and Russell (2001a,b, 2005)'s online boosting works in a similar way to online bagging, but it uses a $Poisson(\lambda)$ distribution, as shown in algorithm 2.2. The parameter λ associated to a training example d is increased when presented to the next ensemble member if the current ensemble member misclassifies the example (line 19). Otherwise, it is decreased (line 15).

In this way, λ has the same role as the weight of the example d in AdaBoost (Freund and Schapire; 1996a, 1997): the examples misclassified by one stage (one classifier) are given half the total weight in the next stage and the correctly classified examples are given the remaining half of the weight. This can be seen by the following (Oza and Russell; 2001b). According to algorithm 2.2, λ_m^{sc} and λ_m^{sw} are the total weights of correctly and incorrectly classified examples for each base model m. We want to find the factors f_m^c and f_m^w that scale λ_m^{sc} and λ_m^{sw} to half the total weight:

[2] A lossless online learning algorithm returns a hypothesis identical to that returned by the corresponding offline algorithm.

Algorithm 2.2 Oza and Russell (2001a,b, 2005)'s Online Boosting Algorithm

Inputs: ensemble $\mathbf{h}$, ensemble size M, training example $d = (\mathbf{x}, y)$ and online learning algorithm for the ensemble members $OnlineBaseLearningAlg$.

1: **if** d is the first training example **then**
2: **for** $m \leftarrow 1$ to M **do**
3: static $\lambda_m^{sc} \leftarrow \lambda_m^{sw} \leftarrow 0$ /* initialize the total weights of correctly and incorrectly classified examples for each base model */
4: **end for**
5: **end if**
6: $\lambda \leftarrow 1$
7: **for** $m \leftarrow 1$ to M **do**
8: $W \leftarrow Poisson(\lambda)$
9: **for** $w \leftarrow 1$ to W **do**
10: $h_m \leftarrow OnlineBaseLearningAlg(h_m, d)$
11: **end for**
12: **if** $h_m(\mathbf{x}) == y$ **then**
13: $\lambda_m^{sc} \leftarrow \lambda_m^{sc} + \lambda$
14: $\xi_m \leftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sc}+\lambda_m^{sw}}$
15: $\lambda \leftarrow \lambda \left(\frac{1}{2(1-\xi_m)}\right)$
16: **else**
17: $\lambda_m^{sw} \leftarrow \lambda_m^{sw} + \lambda$
18: $\xi_m \leftarrow \frac{\lambda_m^{sw}}{\lambda_m^{sc}+\lambda_m^{sw}}$
19: $\lambda \leftarrow \lambda \left(\frac{1}{2\xi_m}\right)$
20: **end if**
21: **end for**

Output: updated ensemble $\mathbf{h}$.

$$\lambda_m^{sc} f_m^c = \frac{\lambda_m^{sc} + \lambda_m^{sw}}{2} \Rightarrow f_m^c = \frac{\lambda_m^{sc} + \lambda_m^{sw}}{2\lambda_m^{sc}} = \frac{1}{2(\frac{\lambda_m^{sc}}{\lambda_m^{sc}+\lambda_m^{sw}})} = \frac{1}{2(1-\xi_m)}$$

$$\lambda_m^{sw} f_m^w = \frac{\lambda_m^{sc} + \lambda_m^{sw}}{2} \Rightarrow f_m^w = \frac{\lambda_m^{sc} + \lambda_m^{sw}}{2\lambda_m^{sw}} = \frac{1}{2(\frac{\lambda_m^{sw}}{\lambda_m^{sc}+\lambda_m^{sw}})} = \frac{1}{2\xi_m}$$

As we can see, these are exactly the factors used in lines 19 and 15 of algorithm 2.2.

To use the system, the classification made by the whole ensemble is by weighted majority vote, with weights based on the accuracy of the ensemble members.

Fern and Givan (2000, 2003)'s online boosting operates in a similar way. However, the weights associated to the training examples are based on Arc-x4 (Breiman; 1996b), instead of AdaBoost. They are calculated as $1 + m_t^4$, where m_t is the number of previous hypotheses that misclassify the new example. This weighting scheme results in a boosting-style algorithm that

emphasizes examples misclassified by many previous hypotheses.

2.5 Online Approaches to Learn in the Presence of Concept Drift

The online learning algorithms which handle concept drift can be divided into two groups: approaches which use a mechanism to detect drifts (Baena-García et al.; 2006; Gama et al.; 2004; Nishida and Yamauchi; 2007a,b; Nishida; 2008) and approaches which do not explicitly detect drifts (Stanley; 2003; Kolter and Maloof; 2003, 2005). Both of them usually handle drifts based directly or indirectly on the accuracy of the current classifiers.

The former approaches use some measure related to the accuracy to detect drifts. They usually discard the current system and create a new one after a drift is detected and/or confirmed. In this way, they can have quick response to drifts. However, these approaches can suffer from non accurate drift detections. The latter approaches usually associate weights to each base learner based on its accuracy, possibly allowing pruning and addition of new classifiers. As there is no clear cut way to define when a drift occurred, they are likely to maintain ensemble members which do not reflect so well the new concept with weights that are affected by the old concept, taking longer time to recover from drifts.

None of the approaches investigates whether it is possible to use information from the old concept in order to aid the learning of the new concept. Besides, even though there are approaches using ensembles to deal with drifts, there is no deep study of why they can be helpful for that and which of their features can contribute or not for dealing with concept drift. Such a study is presented in this thesis and reveals how to use diversity to better deal with concept drift. None of the current approaches in the literature uses the full potential of diversity for dealing with drifts revealed by the study.

The next sections present some existing approaches and their positive and negative points in more detail. The approaches analysed are: Drift Detection Method (Gama et al.; 2004) (section 2.5.1), Early Drift Detection Method (Baena-García et al.; 2006) (section 2.5.1), Statistical Test of Equal Proportions (Nishida and Yamauchi; 2007b) (section 2.5.1), Two Online Classifiers For Learning And Detecting Concept Drift (Nishida; 2008) (section 2.5.2), Concept Drift Committee (Stanley; 2003) (section 2.5.3), Dynamic Weight Majority (Kolter and Maloof; 2003) (section 2.5.4) and Addictive Expert Ensembles (Kolter and Maloof; 2005) (section 2.5.4).

2.5.1 Methods which Reset the System Upon Drift Detection

Some of the approaches which reset the system upon drift detection are: Drift Detection Method (DDM), Early Drift Detection Method (EDDM) and Statistical Test of Equal Proportions

(STEPD).

DDM (Gama et al.; 2004) is based on the idea that the error rate of a learning algorithm should decrease as the number of training examples increases when the distribution of the examples is stationary. So, a significant increase in the error of the algorithm suggests that the class distribution is changing. It is important to note that ensemble approaches that do not explicitly detect drifts, but handle them by pruning ensemble members with high training error, are also based on this idea.

DDM stores the minimum error rate (p_{min}) and standard deviation (s_{min}) obtained so far. If $p_i + s_i >= p_{min} + 2s_{min}$, where p_i is the current error rate and s_i is the current standard deviation, a warning level is triggered. If $p_i + s_i >= p_{min} + 3s_{min}$, it is considered that there is concept drift.

EDDM (Baena-García et al.; 2006) is inspired on DDM. However, it is based on the idea that not only the accuracy is improved while the learning of a stable concept is occurring, but also the distance between two errors increases. So, the average distance between two errors (p'_i) and its standard deviation (s'_i) is calculated. Their maximum values so far are stored (p'_{max} and s'_{max}). If $(p'_i + 2s'_i)/(p'_{max} + 2s'_{max}) < \alpha$, a warning level is triggered. If $(p'_i + 2s'_i)/(p'_{max} + 2s'_{max}) < \beta$, $\alpha > \beta$, it is considered that there is concept drift.

STEPD (Nishida and Yamauchi; 2007b) assumes that, after a concept drift, the difference between the accuracy for recent examples and the overall accuracy from the beginning of the learning should be statistically different. So, STEPD compares these two accuracies by using a statistical test of equal proportions. The proportions are $\hat{p}_A = r/(n - W)$ and $\hat{p}_B = s/W$, where n is the total number of examples learnt by the online classifier, s is the number of correct classifications among the most recent W examples and r is the number of correct classifications among $n - W$ examples (excluding the most recent W examples from the n examples). A drift can be detected only after $n \geq 2W$. The statistic is calculated by using the Pearson's chi-square test with Yates' correction for continuity (Yates; 1934), which does not have high computational costs. If $\hat{p}_A > \hat{p}_B$ and a statistically significant difference is detected with a certain one-tailed significance level, a significant decrease in the recent accuracy is detected. STEPD uses two different significance levels: α_w (warning) and α_d (drift). While $\hat{p}_A > \hat{p}_B$ and there is a statistically significant difference using α_w, a warning level is set. When $\hat{p}_A > \hat{p}_B$ and there is a statistically significant difference using α_d, it is considered that there is a concept drift.

All these methods start storing the new incoming examples when a warning level is triggered. Then, when the drift is confirmed, the model learnt so far (including the variables stored by the drift detection method, such as p_{min}, s_{min}, p'_{max}, s'_{max}, n) is reset and a new model starts being learnt, considering the examples stored since the warning level was triggered. In EDDM, new p'_{max} and s'_{max} start being considered after a drift detection only after 30 errors have happened. If the similarity between $(p'_i + 2s'_i)$ and $(p'_{max} + 2s'_{max})$ starts to increase after a warning level is

triggered, the stored examples are removed and the method returns to normality.

Successful points

These approaches can quickly react to drifts once they are detected.

Experiments using perceptrons, neural networks and decision trees as the base learners on eight artificial databases presenting concept drift and one real world electricity market database showed that DDM recovered faster from concept drift than base learners not using DDM, usually achieving significantly better generalization in the end of the training.

Experiments on an artificial database presenting a very slow gradual drift showed that EDDM reacted faster than DDM for this type of drift. Similar behaviour occurred for a real world database. Experiments on four artificial databases with gradual and abrupt concept drifts showed that this approach had similar behaviour to DDM for these types of drifts. The experiments used three different base learners: C4.5 decision trees (Quinlan; 1993), instance-based learning (Aha et al.; 1991) and nearest-neighbourhood with generalization (Martin; 1995).

Experiments using instance-based learning (Aha et al.; 1991) and naive-bayes as the base learners on five artificial data sets presenting different types of drifts showed that STEPD usually detected drifts faster than DDM and EDDM, obtaining better error rate for sudden drifts and comparable error rate to EDDM on gradual drifts. However, the speed of the detection in comparison to EDDM is questionable, as a single choice of parameters was tried, when EDDM actually has parameters which allow tuning the trade-off between speed of drift detection and false alarms.

Problems

As the model is reset whenever a concept drift is detected, these approaches cannot use any information previously learnt. So, they can present higher increase in the generalization error right after a drift in comparison to approaches which do not handle drifts, especially when the drift does not cause many changes. Besides, no advantage can be taken from the previous model in the case of recurrent drifts. Resetting the model after a drift detection also makes the approaches very sensitive to false alarms.

Another problem is that, if the warning level is triggered and there is really a drift at that moment, the accuracy of the classifier for the new concept can be improved only after the drift level is triggered.

Moreover, when the warning level is triggered, it is necessary to store the new incoming examples. This is not considered a true online learning behaviour. However, a modification in the implementation to start training a new classifier with the new examples when a warning level is triggered would transform the approaches in true online learning approaches.

DDM may also suffer from slow drift detection, as it detects drifts based on the current error rate (p_i) and its standard deviation (s_i), which are calculated considering values since the last detected concept drift. So, it may take a long time for $p_i + s_i$ to considerably increase when a drift happens and result in a drift detection. A similar problem may happen to EDDM, even though the parameter β allows tuning the trade-off between speed of drift detection and false alarms. STEPD has a parameter W which works in a similar way to a sliding window to detect drifts. A too large W will take longer to detect drifts and a too small W may cause unstable behaviour. Besides, STEPD needs at least $2W$ examples between consecutive drifts to detected them properly. So, drifts are not always continuously monitored.

2.5.2 Two Online Classifiers for Learning and Detecting Concept Drift (Todi)

Todi (Nishida; 2008) uses two online classifiers for learning and detecting drifts. One of them (H_0) is rebuilt every time a concept drift is detected. The other one (H_1) is not rebuilt when a drift is detected, but can be replaced by the current H_0 if a drift is confirmed. Similarly to STEPD, Todi detects concept drift by performing a statistical test of equal proportions to compare H_0's accuracies on the most recent W training examples and on all the training examples presented so far excluding the last W training examples. H_0 must have been trained with at least $2W$ examples for the drift to be detected.

The authors claim that the approach uses only one significance level α. They argue that it is not necessary to have a warning level to store training examples in a short time memory, as DDM, EDDM and STEPD, because H_1 is maintained after the concept drift detection. So, the accuracy of the ensemble is less deteriorated in the case of false alarms. However, Todi actually also works with two levels. Instead of calling them "warning" and "drift" levels, they are called "drift detection" and "drift confirmation".

After the detection of a concept drift, a statistical test of equal proportions with significance level β is done to compare the number of correctly classified training examples by H_0 and H_1 since the beginning of the training of H_0. If statistically significant difference is detected, which means that H_0 was successful to handle concept drift significantly, the drift is confirmed. Then, H_0 replaces H_1 and a new H_0 starts learning from scratch. If the concept drift is not confirmed, Todi reinitializes H_0 and keeps H_1.

The classification is done by selecting the output of the most accurate classifier considering the W most recent training examples: considering that s_i is the number of correct classifications of the classifier H_i for the last W examples, if $s_0 > s_1 + 1$, Todi selects the classifier H_0 to classify the instance presented in the current time step. If $s_0 < s_1 - 1$, then Todi selects H_1. Otherwise, Todi selects the classifier that was selected in the previous time step.

Successful points

Todi was able to confirm drifts without misdetections and to obtain better accuracy than DDM and STEPD using naive-bayes as the base learners for the SEA Concepts (Street and Kim; 2001) data set using both sudden (original data set) and gradual (1000 time steps to completely change from the old to the new concept) drifts. Besides, Todi's error rate was better than Bogofilter's (Raymond et al.; 2007), which is a well-known spam filter, and STEPD's on a real world spam data set.

Problems

Even though the authors claim that Todi does not need a warning level, we can consider their "drift detection" as the warning level used by DDM, EDDM and STEPD and their "drift confirmation" as the drift detection level. As explained in section 2.5.1, DDM, EDDM and STEPD can be considered as true online learning algorithms if, instead of storing the examples seen during a warning level in a memory for posterior training, these examples are directly used to train a new classifier. Todi works in a similar way to that. The main difference is that the new classifier can also be used for predictions before a drift is confirmed and that concept drift is monitored by using a classifier that is built only with examples learnt from the moment in which a drift is confirmed.

Todi uses a window over data which may create an unstable system if it is too small and may not react quickly to drifts if it is too big. Besides, subsequent drifts can only be detected if there are at least $2W$ examples between them. So, drifts cannot be always continuously monitored.

Although H_1 is not rebuilt when a drift is detected (only after a drift is confirmed), it is not prepared to deal with new concepts. So, even though it may increase the robustness to false alarms, it cannot help to reduce the drop in generalization right after the drift if the concept really changed.

The approach also has the problem of presenting no mechanism to deal with recurrent concepts.

2.5.3 Concept Drift Committee (CDC)

In this approach (Stanley; 2003), an ensemble (committee) of decision trees is created to handle concept drift. All the ensemble members are trained with incoming training examples. The ensemble is initially empty and, every time that a new training example is made available, a new decision tree is added to the ensemble if the maximum ensemble size was not attained yet. When the maximum size is achieved, a new decision tree is added only if an existing decision tree can be removed from the ensemble. Each ensemble member has a weight proportional to the accuracy on the last n training examples. The decision tree with lowest weight below some

threshold t is eliminated. Every ensemble member also has an age (number of time steps since it was added to the ensemble) and cannot be eliminated or used for classification before an age of maturity is reached. So, ensemble members that were not trained enough are given a chance to learn the concept without prejudicing the ensemble's generalization.

Successful points

The approach showed to have comparable generalization to FLORA4 (Widmer and Kubat; 1996) on the first sudden drift using the STAGGER concepts data set (Schlimmer and Granger; 1986) and better generalization on the second sudden drift (although this difference may be caused by the use of different base learners). Both approaches were better than instance-based learning 3 (IB3) (Aha et al.; 1991).

The approach showed to have comparable generalization to FLORA4 (Widmer and Kubat; 1996) on a boolean data set containing drifts with drifting time (number of time steps to complete the drift) 100, but better generalization when the drift had drifting time 200.

Problems

CDC had worse generalization than FLORA4 during the first concept of the STAGGER data set.

In the beginning of the learning, a new decision tree is added at each time step while the ensemble does not have the maximum number of decision trees. As diversity is not encouraged, very similar decision trees will be created. So, the memory requirements and computational time are increased unnecessarily. The accuracy of the system during this period is likely to be similar to the accuracy of a single decision tree, wasting the power of ensembles to achieve better accuracy.

The approach does not present any specific method to handle recurrent concepts. The elimination of ensemble members with low accuracy on the most recent n training examples may remove members that could be useful in the case of a recurrent concept.

The size of the window n critically influences the behaviour of the approach. A too large n would reduce the accuracy to handle drifts, whereas a too small n may create an unstable system.

Moreover, no strategy to improve the learning of existing members on the new concept is adopted, so that the ensemble only recovers from a drift when the new ensemble members achieve the age of maturity and are accurate enough. So, there is a delay in the system's reaction to drifts. This is probably one of the reasons why CDC does not achieve very good accuracy for abrupt drifts.

2.5.4 Dynamic Weight Majority (DWM) and Addictive Expert Ensembles (AddExp)

Kolter and Maloof (2003, 2007) proposed DWM to track concept drift in online problems. DWM is one of the most cited online learning approaches to handle drifts. Each ensemble member has a weight that starts with value 1 and is reduced by a multiplicative constant β ($0 \leq \beta < 1$) when it makes a wrong prediction, similarly to Littlestone and Warmuth (1994)'s Weighted Majority. However, the weight of an ensemble member can be updated only in time steps multiple of p, where p is a pre-defined value.

Besides, at every p training examples, (1) a new classifier is added if the last training example was misclassified by the ensemble after the weight updates and (2) all ensemble members with weight less than a pre-defined threshold are removed. So, when the target concept changes, (1) new ensemble members can be created to learn a new concept without having to first forget the old one and (2) poorly performing members which would take a long time to forget a concept previously learnt can be removed. As all existing members are trained with new data, ensemble members with not so low weight are given the opportunity to try to forget old concepts and learn the new one. At every p training examples, the weights of all ensemble members are normalized before adding any new member to the ensemble, so that the new member to be included does not dominate the decision making of all the others.

A similar method, AddExp, was proposed by Kolter and Maloof (2005). In this method, a new classifier is always added when the prediction of the ensemble as a whole is wrong. The weight of this classifier is equal to the sum of the weights of all the existing ensemble members multiplied by a constant γ ($0 \leq \gamma \leq 1$). The weight of an ensemble member is multiplied by a constant β ($0 \leq \beta < 1$) at every time step in which it gives a wrong prediction.

If no pruning is adopted, mistake and loss bounds can be defined for AddExp. However, without pruning, the ensemble size can become extremely large and impractical. So, a maximum ensemble size can be used. When the maximum is achieved, either the oldest or the lowest weight ensemble member can be deleted before adding a new member. The first pruning strategy does not affect the bounds defined to AddExp without pruning. However, the authors recognize that this strategy may be impractical, as an extremely large ensemble size might be necessary in order to keep the bounds. Besides, the elimination of the oldest ensemble member would not be the best alternative during periods of stability. The second strategy works well in practice, but has no theoretical guarantee.

Successful points

The base learners are trained with different sequences of data, as the ensemble members are created in different moments. So, diversity does not need to be built *a priori*.

Experiments using naive bayes and online decision trees on the STAGGER Concepts (Schlimmer and Granger; 1986) showed that DWM recovered much faster from drifts than the single classifiers, achieving better generalization. Besides, DWM performed almost as well as the single classifiers trained on each target concept individually (classifiers with perfect forgetting). Similar behaviour was observed through experiments on the SEA Concepts (Street and Kim; 2001) using naive bayes. A comparison to Blum (1997)'s implementation of Weighted Majority showed that DWM had comparable or better generalization when the target concept changed. DWM using online decision trees also outperformed four other approaches specifically designed to concept drifting after the first drift. AddExp had comparable accuracy to DWM for the STAGGER data set and better accuracy for a moving hyperplane data set.

Problems

For DWM, the inclusion/elimination of classifiers only at every p training examples makes it not possible to handle new drifts during the period p. This problem can be overcome if we set $p = 1$, even though such a setting can increase the training time considerably. Another problem is that the pruning mechanism used by DWM provides no guarantee as to the number of experts created. Moreover, the fact that the newest ensemble member is chosen to have the highest weight in the ensemble makes the newest member become dominant too quickly (even with the weights normalization), which can be harmful particularly in the presence of noise (Kolter and Maloof; 2005).

AddExp can be considered more robust to noise when pruning is adopted than DWM. However, as it adds a new classifier at every example misclassified by the ensemble, the training can become prohibitively slow when there is a concept drift and the accuracy of the system drops. This problem happens even if pruning is used, because the rate of inclusion/elimination of learners can become very high. DWM manages to reduce the slowness when larger p values are used.

Neither DWM nor AddExp present any specific method to deal with recurrent concepts. As the weights can only reduce, but not increase, it is not possible for an old ensemble member to increase the weight when there are recurrent concepts. They have to wait for the other members to have their weights considerably reduced. However, the approaches can still somewhat benefit from the old ensembles if their weights are not much reduced so as to be deleted or to take too long for other members to get even lower weights when the concept reappears.

Another problem is that no strategy to improve the learning of existing members on the new concept is adopted. The old classifiers maintained in the ensemble are simply used for predicting the new concept. As it is shown in section 5.3, a classifier which learnt an old concept well is not likely to present good convergence to the new concept even if the drift is not so severe.

The experiments performed with DWM showed that it usually did not perform so well

during the first concept. AddExp was not compared to so many approaches as DWM.

2.6 Summary and Discussion

This chapter explains that there are many incremental learning approaches in the literature, but they all present the problem of determining the chunk size, which is particularly critical when modelling changing environments. Besides, no drift can be detected before a whole new chunk of data is received. Incremental approaches also usually lack stability during stable concepts, suffer from delayed system update and have higher memory and time requirements. Considering these reasons and the fact that online learning approaches can be used for both online and incremental learning, the thesis concentrates on online learning.

When analysing online learning approaches for stable concepts, we can observe that sequential-generation approaches present the problem of having to determine when to stop the training of a particular ensemble member and start the training of the other. Single-update approaches do not use the full power of ensembles, which can obtain faster convergence when allowing many members to learn a particular training example. When working in the presence of concept drift, these problems are even more critical if no additional strategy to handle drifts is adopted.

Parallel-generation multiple-update approaches can be further divided into approaches which send the same sequence or a different sequence of training examples to each ensemble member. As the former approaches need diversity to be built *a priori*, it is preferable to use the latter. Even though parallel-generation multiple-update approaches are more recommendable for online learning, they also present problems to recover from drifts if no additional strategy is adopted.

The existing online learning approaches which attempt to handle concept drift can be divided into approaches which explicitly or implicitly detect drifts. The former approaches use some measure related to the accuracy to detect drifts. They can have quick response to drifts as long as the drifts are detected early. However, they suffer from non accurate drift detections. The latter approaches do not have a clear cut to conclude that a drift occurred in a particular moment, being likely to take longer time to recover from drifts for maintaining base learners which learnt old concepts.

None of the existing online learning approaches exploit information learnt from the old concept in such a way to aid the learning of the new concept. Besides, even though ensembles have been used to deal with concept drifts, there is no deep study of why they can be helpful for that and which of their features can contribute or not to deal with concept drift. Such a study is presented in this thesis, revealing how to use diversity to better deal with concept drift. None of the existing approaches fully uses diversity in this way to better deal with concept drift. The thesis presents strategies to use diversity and information learnt from the old concept in order

to aid the learning of the new concept. A new approach which uses these strategies is proposed in chapter 6.

This chapter also explains NCL, an approach that has been successfully used for offline learning, but is not exploited for online and incremental learning in the literature. Such a study is presented in chapter 3. Previous work on diversity in offline ensemble learning is also presented. No previous work on diversity in changeable environments can be found in the literature. Such a work is presented in chapter 5.

Chapter 3

Negative Correlation in Incremental and Online Learning

As explained in section 2.2, Negative Correlation Learning (NCL) (Liu and Yao; 1999b,a) is an ensemble learning approach which directly encourages diversity through the use of a penalty term in the error function of the neural networks belonging to the ensemble. It has shown to outperform other ensemble methods in offline mode (Islam et al.; 2003; Wang et al.; 2004; Chandra and Yao; 2006). So, it is important to analyse whether we can also benefit from it in online or incremental environments. In particular, as NCL has a penalty term which allows encouraging more or less diversity, we should check whether this approach can be used for analysing the impact of diversity in online learning in the presence of concept drift, for answering research question 1.2.2.

Sections 3.1 and 3.2 analyse NCL in incremental and online learning, respectively, without considering concept drift. They provide an answer to research question 1.2.1 and suggest not to use NCL for the study of diversity performed for answering research question 1.2.2. Section 3.1 also illustrates one of the problems presented by many of the incremental learning approaches.

3.1 Negative Correlation Approaches for Incremental Learning

This section presents an extensive analysis of incremental NCL, aiming at determining and discussing the strong and weak points of NCL in this learning scenario. Two different approaches (section 3.1.1) to perform NCL are analysed: Fixed Size NCL and Growing (Size) NCL. The data sets and experimental design are explained in section 3.1.2. The analysis itself is presented in section 3.1.3 and checks the following points:

1. Whether negative correlation can achieve good generalisation in incremental learning (section 3.1.3.1 and 3.1.3.5).

2. Whether negative correlation improves generalisation when new data chunks are received from the incremental environment (section 3.1.3.2).

3. The behaviour of negative correlation to forgetting knowledge previously learnt in incremental learning (section 3.1.3.3).

4. The capacity that some ensemble members have to adapt better than the others for new data, which is one of the ensemble properties which can help incremental learning if well exploited (section 3.1.3.4).

It provides the incremental learning part of the answer to the research question presented in section 1.2.2: "what are the strengths and weaknesses of negative correlation in incremental and online learning?"

This section shows that NCL is promising in incremental learning. It manages to improve generalisation from the first to the last chunk of data and, depending on the strategy adopted, can benefit from several ensemble members which adapt differently for new data or overcome forgetting. However, overcoming forgetting so as to achieve higher generalisation is not straightforward. Section 3.1.4 presents a third strategy which usually maintains or improves accuracy in relation to the other two strategies analysed, at the same time it reduces forgetting in comparison to one of them. The results are encouraging and suggest that further studies combining the strengths of the approaches may achieve higher generalisation.

3.1.1 Fixed Size and Growing NCL

Two approaches to perform negative correlation in incremental learning are introduced and analysed: Fixed Size NCL and Growing (Size) NCL.

Fixed Size NCL is a direct way of performing NCL in incremental environments. In this approach, an ensemble of neural networks is created and trained with the first available data chunk. When another chunk arrives, the whole ensemble is further trained on the new data, using the weights learnt from the previous chunk as initial weights. No training is performed with old data.

This approach has the advantage that it allows all the networks to learn the incoming data. In this way, some of the neural networks can adapt faster and better than others and it is expected that these neural networks help the ensemble to overcome the problem of the networks with bad adaptation to the new data. Besides, it is believed that the property of a single training example to contribute to the training of many ensemble members is essential for obtaining a fast convergence to the desired target concept (Fern and Givan; 2003). Moreover, as the entire ensemble is always used to all incoming data, if there are similarities among new data and old data, the initial training error on the new data can be lower. The problem of Fixed Size NCL

is that the neural networks can forget information previously learnt, situation that we will refer to here as "forgetting".

In Growing NCL, an ensemble is created with initially only one neural network. The network is trained with the first available data chunk. Then, a new network is inserted in the ensemble for each new chunk. Only the new neural network is trained with the new chunk. The previous networks are not trained on new data, but their outputs are calculated in order to interact with the new network using NCL.

This approach has the advantage that it can tackle the problem of forgetting information previously learnt, as each neural network is trained with only one incoming data set. As only one neural network is trained with each new chunk, instead of the whole ensemble, its learning is faster than Fixed Size NCL. However, it does not benefit from the increased accuracy that the training of several learners can provide when using ensembles.

The advantages and disadvantages of the approaches are further discussed in section 3.1.3.

3.1.2 Experimental Design

As explained in the beginning of section 3.1, the experiments aim at determining and discussing the strong and weak points of NCL in incremental learning through an extensive analysis of the two approaches described in section 3.1.1. The four points explained in the beginning of section 3.1 are analysed and the measures used for the analysis are explained in the respective sections, in order to facilitate understanding.

To support the analysis, NCL is compared with SONG (Inoue and Narihisa; 2003) (section 2.3.2), which is a successful approach to incremental learning that joins the advantages of a base classifier which is proper to incremental learning to the advantages of using an ensemble to perform incremental learning. NCL is also compared with single MLPs, Learn++ (Polikar et al.; 2001) (section 2.3.1) and the evolutionary approach introduced by Seipone and Bullinaria (2005) (section 2.3.3).

The results obtained by Learn++ and Seipone and Bullinaria (2005)'s evolutionary approach were obtained from the literature (Polikar et al.; 2001; Seipone and Bullinaria; 2005). For the other approaches, thirty executions were done according to table 3.1, for each data set. The neural networks used with NCL are MLPs trained with stochastic backpropagation. The criteria used to stop the learning are:

- Early stop based on the Generalisation Loss (GL) (Prechelt; 1994). According to this criterion, if the generalisation loss (based on the validation error) is higher than a predefined parameter α, the training stops. This criterion is considered only after a certain training progress (defined in (Prechelt; 1994)) is attained. A third of each training chunk was used as validation data for the MLPs, instead of being used for training.

- Maximum number of epochs.

Table 3.1: Approaches Used in the Experiments for the Study of Incremental NCL.

Single MLP	h hidden nodes
	$3h$ hidden nodes
	$5h$ hidden nodes
	$10h$ hidden nodes
Fixed Size NCL	5 MLPs with h hidden nodes each
	10 MLPs with h hidden nodes each
Growing NCL	MLPs contain h hidden nodes each
SONG	1 SGNT
	5 SGNTs
	10 SGNTs

Table 3.2 summarizes some of the parameters used in the executions performed with NCL and single MLPs. It shows the number h of hidden nodes, the maximum number of epochs, the value α to the GL criterion and the minimum progress. All the parameters were empirically chosen, based on preliminary executions. The MLPs were trained with learning rate of 0.1, except for Mushroom, in which the learning rate was 0.05. The interval of initial values for the weights was $[-1, 1]$ and the strip size, used by the GL and training progress criteria, was 5, except for Vehicle, in which it was 10. The strength parameter γ for the NCL penalty term was 0.390625. This value corresponds to $\lambda = 0.75$ when 25 MLPs are used as the base learners. The single MLPs were executed with both h, $3h$, $5h$ and $10h$ hidden nodes, although the use of $10h$ hidden nodes may provide a fairer comparison with ensembles. The executions with SONG used the parameter $\alpha = 1.0$.

Table 3.2: NCL Parameters for the Study of Incremental NCL.

Database	Hidden Nodes	Max. Epochs	GL_α	Min. Progress
Letter	40	300	1.5	10
Vehicle	30	2000	5	50
Optdigits	10	100	100	0
Adult	5	100	5	20
Mushroom	25	0.05	50	0

The experiments used five benchmark classification databases from the UCI Machine Learning Repository (Newman et al.; 2010): Letter, Vehicle, Optical Digits, Adult and Mushroom. Each data set was divided to generate several training data chunks and one test set. There is no (known) concept drift in these databases. Table 3.3 presents a summary of them. It shows the number of input and output attributes, the number of patterns of the database, the size (and number) of the training chunks used in the incremental learning and the size of the test data sets. The signal $\sim$ indicates that some of the training chunks contain one pattern less than the number indicated.

The Vehicle data set is considered as one of the most difficult databases in the repository,

Table 3.3: Databases for the Study of Incremental NCL.

Database	Inputs	Outputs	Patterns	Training	Test
Letter	16	26	20000	2000 (9)	2000
Vehicle	18	4	846	210 (3)	216
Optdigits	64	10	5620	200 (6)	4420
Adult	14	2	45222	$\sim$ 4523 (9)	4523
Mushroom	21	2	8124	$\sim$ 813 (9)	813

since generalisation performances using various algorithms have been in the 65%-80% range (Polikar et al.; 2001). MLPs usually do not attain good test error rates for Letter, while methods like K-Nearest Neighbors (Larose; 2004) manage to attain good generalisation (Adamczak et al.; 1997). Letter, Mushroom and Adult are databases with more than 8000 patterns. So, it was possible to divide these databases in various different training chunks. The sizes of the training chunks and test sets for Vehicle and Optical Digits were chosen in order to allow comparisons with the results published in the literature for Learn++ (Polikar et al.; 2001) and Seipone and Bullinaria (2005)'s evolutionary approach.

3.1.3 Analysis

This section presents the analysis, considering the four points explained in the beginning of this section and aiming at identifying and discussing the positive and negative points of NCL in incremental learning. The analysis shows that it is possible to use negative correlation in incremental learning, although each approach analysed also has its weakness.

It shows that some of the networks of the Fixed Size NCL ensemble can adapt better than the others, and they are not the same for different data chunks. This is a useful feature for dealing with incremental learning, as explained in chapter 1. However, this approach suffers more from forgetting than Growing NCL. Growing NCL can be used to overcome forgetting, but does not take advantage of using more than one neural network to learn new data, getting worse generalisation. The improvement in generalisation after training with new data for both the approaches is comparable with other approaches specifically developed for incremental learning. The study reveals encouraging results and shows that NCL is a promising approach to incremental learning. One of the possible ways to further improve NCL in incremental learning is to adopt a selective approach, as explained in section 3.1.4.

Section 3.1.3.1 presents an analysis of the generalisation obtained by the NCL incremental approaches, SONG and single MLPs. Section 3.1.3.2 presents an analysis of the improvement in generalisation as more data chunks arrive. Section 3.1.3.3 presents an analysis of forgetting. Section 3.1.3.4 presents an analysis of the differences in the adaptation to new data chunks presented by different ensemble members. Section 3.1.3.5 presents a comparison between NCL, Learn++ and Seipone and Bullinaria (2005)'s evolutionary approach.

3.1.3.1 Generalisation

The generalisation (1 - test classification error) on the test set was measured after the presentation and learning of each training chunk. We will call the presentation and learning of each chunk as an incremental step. T-Student statistical tests (Witten and Frank; 2000) with level of significance of 5% and 1% were used to check whether there is statistically significant difference between the average generalisation of the approaches in each incremental step. When the difference between two averages is statistically significant and one of the averages is better/worse than the other, we will simply refer to it as being statistically better/worse than the other, to simplify the writing.

Usually, in the literature, level of significance of 1% is used for critical applications, while 5% is used for non-critical applications. However, Dietterich (1998) concluded that T-Student tests with 5% of significance has high probability of incorrectly detecting that there is difference when no difference exists, not being recommended. So, we perform tests with level of significance of 1% to reduce the probability of this problem and consider the results with this level as more plausible.

In this section, it is considered that the approach A is statistically better than the approach B for a particular database if A contains more incremental steps with average generalisation statistically better (higher) than B, than B has in relation to A. It is possible for us to consider that here because there is no occurrence of an approach B being much better than A in very few time steps, but just slightly worse in all the others. If that occurred, adopting such a procedure to consider A better than B might be questionable for ignoring the magnitude of the generalisation.

The Influence of the Number of Ensemble Members and Hidden Nodes

We observe in this section that a higher number of ensemble members or hidden nodes tends to increase generalisation. SONG with 5 SGNTs was always statistically worse than SONG with 10 SGNTs and the difference is statistically significant when 5% of level of significance is adopted. When 1% is adopted, some of the differences become equalities, as expected, reducing the probability of detecting differences that do not exist. SONG with 1 SGNT was always statistically worse than SONG with 10 SGNTs.

Fixed Size NCL with 5 MLPs was always statistically equal or worse than Fixed Size NCL with 10 MLPs when both 5% and 1% of level of significance are adopted. Single MLPs with h, $3h$ and $5h$ hidden nodes were either statistically equal or worse than Fixed Size NCL with 10 MLPs when using level of significance of 1%. When using 5%, single MLPs with $5h$ were statistically better for Adult.

So, from this point onwards, the analysis will always consider the following approaches:

- Single MLP with $10h$ hidden nodes;
- Fixed Size NCL with 10 MLPs, with h hidden nodes each;
- Growing NCL in which the MLPs contain h hidden nodes each;
- SONG with 1 SGNTs.
- SONG with 10 SGNTs.

The number of hidden nodes of each MLP and the number of MLPs of Fixed Size NCL will be omitted when referring to these approaches.

NCL Approaches vs. SONG

Figure 3.1 shows the average generalisation for each database. For Vehicle and Adult, MLP and the NCL incremental approaches were statistically better than SONG (both 1 and 10 SGNTs). For the other databases, SONG was statistically better. T-Student statistical tests at each incremental step with both 5% and 1% of level of significance confirm this analysis. The p-values of the statistical tests for these comparisons were in general very low. For instance, the p-values for the comparison between Fixed Size NCL and SONG with 10 SGNTs, which are the best NCL and SONG configurations, were always lower than 10^{-10}, apart from the ones for Mushroom, which are shown in table 3.4.

So, NCL is comparable (sometimes statistically better and sometimes statistically worse) than SONG in terms of generalisation. This is important to show the applicability of NCL in incremental learning.

Table 3.4: Study of Incremental NCL – P-values for the T-student Statistical Tests Comparing Fixed Size NCL Using 10 MLPs and SONG Using 10 SGNTs for Mushroom. P-values less than 0.05 and 0.01 indicate a statistically significant difference at the levels of significance of 5% and 1%, respectively.

Incremental Step	P-Value
1	$4.05942 * 10^{-17}$
2	$4.95916 * 10^{-12}$
3	$1.01298 * 10^{-02}$
4	$1.64551 * 10^{-03}$
5	$5.90281 * 10^{-06}$
6	$1.77480 * 10^{-11}$
7	$3.31545 * 10^{-04}$
8	$1.13900 * 10^{-05}$
9	$7.01853 * 10^{-03}$

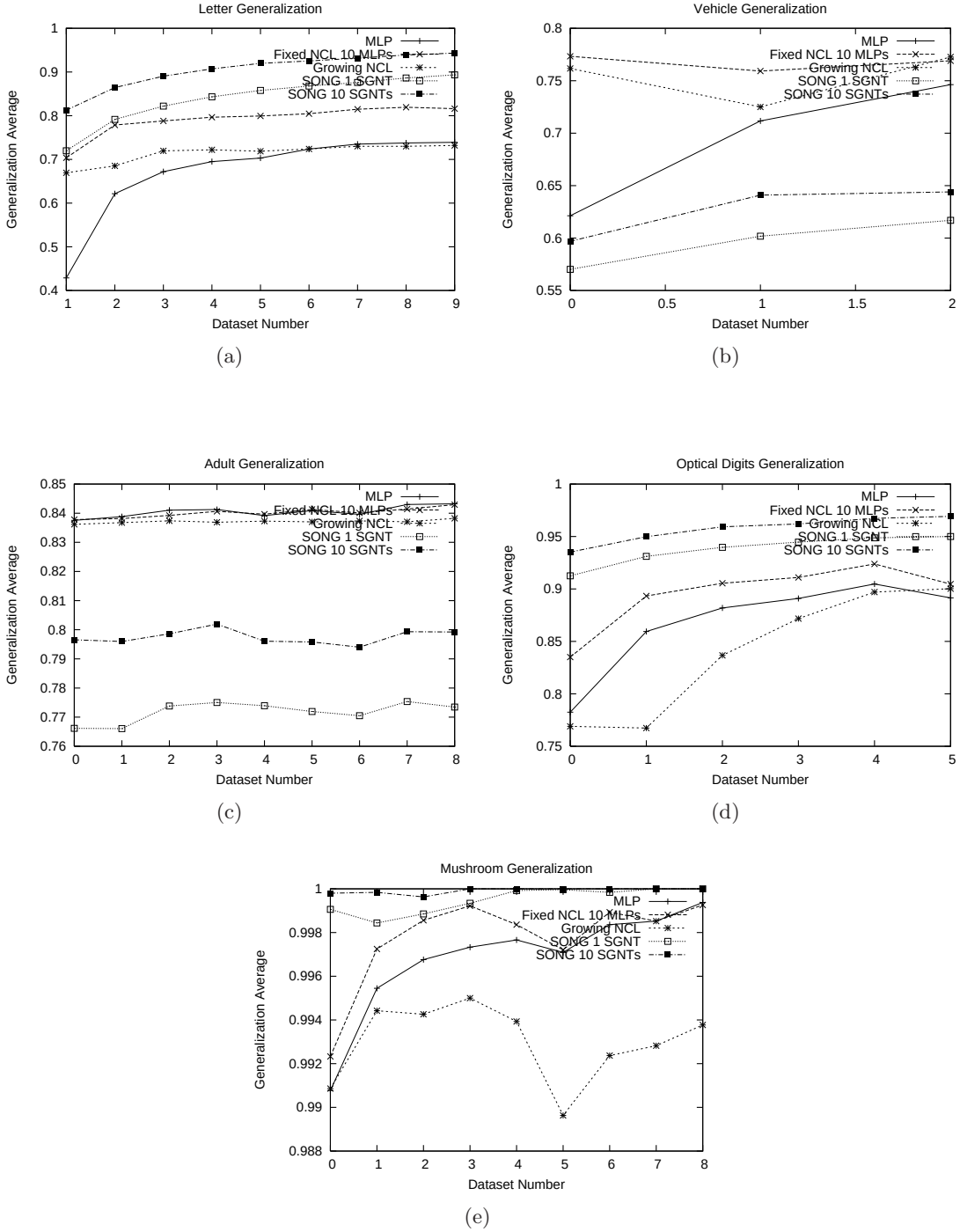


Figure 3.1: Study of Incremental NCL – Average generalisation accuracy.

Fixed Size NCL vs. Growing NCL vs. Single MLP

Table 3.5 shows the rank of difficulty of the databases, from the most difficult to the easiest, based on the generalisation obtained by single MLPs on the last incremental step. The average generalisations of each approach in the first and last incremental steps is also shown.

Table 3.5: Databases Difficulty – From the most difficult to the easiest.

Rank	Database	Generalisation Av. (first–last incremental step)		
		MLP	Fixed Size NCL	Growing NCL
1	Letter	0.42900–0.73920	0.70362–0.81617	0.66928–0.73193
2	Vehicle	0.62130–0.74630	0.77330–0.76898	0.76173–0.77269
3	Adult	0.83754–0.84321	0.83781–0.84294	0.83618–0.83823
4	Optical Digits	0.78250–0.89153	0.83509–0.90464	0.76900–0.90027
5	Mushroom	0.99077–0.99938	0.99233–0.99926	0.99086–0.99377

Considering table 3.5 and the average generalisation (figure 3.1), it is possible to make the following observations about the generalisation of single MLP, Fixed Size NCL and Growing NCL:

- For the most difficult databases (Letter and Vehicle), the best approach was Fixed Size NCL and the worst was MLP.
- For the easiest databases (Optical Digits and Mushroom), the best approach was also Fixed Size NCL. However, the worst was Growing NCL.
- For the Adult database, the best approach was MLP and the worst was Growing NCL. In many of the incremental steps, however, MLP had statistically equal generalisation to Fixed Size NCL.

All these observations are confirmed by T-Student statistical tests with both 5% and 1% of level of significance. For instance, the p-values for the comparisons between Fixed Size NCL, which is usually the best approach, and the other approaches are shown in table 3.6. We can observe that the best approach was usually Fixed Size NCL and that it was statistically better than Growing NCL for all the databases. In section 3.1.3.3, we can see that the training accuracies of the new MLPs are lower than the accuracies of each MLP which compose the Fixed Size NCL. As it is explained in that section, this is likely to be one of the reasons why Growing NCL has worse generalisation than Fixed Size NCL.

Table 3.6: Study of Incremental NCL – P-values of Part of the T-Student Statistical Tests to Compare Generalisation Between Fixed Size NCL, Growing NCL and MLP. P-values less than 0.05 and 0.01 indicate statistically significant difference at the levels of significance of 5% and 1%, respectively.

(a) Fixed Size NCL vs. Growing NCL

Inc. Step	Letter	Vehicle	Adult	Optical Digits	Mushroom
1	$9.66621 * 10^{-08}$	$4.49835 * 10^{-03}$	$7.08818 * 10^{-02}$	$2.96778 * 10^{-14}$	$1.99576 * 10^{-02}$
2	$1.13135 * 10^{-34}$	$6.44818 * 10^{-06}$	$7.23507 * 10^{-02}$	$1.13789 * 10^{-17}$	$5.89796 * 10^{-05}$
3	$2.90407 * 10^{-28}$	$4.27355 * 10^{-01}$	$3.41016 * 10^{-03}$	$4.72308 * 10^{-21}$	$3.21281 * 10^{-08}$
4	$2.16520 * 10^{-22}$		$5.70979 * 10^{-09}$	$5.04222 * 10^{-17}$	$3.02731 * 10^{-10}$
5	$5.97603 * 10^{-26}$		$8.45436 * 10^{-08}$	$3.17242 * 10^{-19}$	$3.40579 * 10^{-08}$
6	$7.53755 * 10^{-28}$		$5.24605 * 10^{-09}$	$2.69334 * 10^{-02}$	$1.78809 * 10^{-07}$
7	$3.72202 * 10^{-29}$		$1.46657 * 10^{-05}$		$1.86148 * 10^{-10}$
8	$1.06238 * 10^{-32}$		$3.88866 * 10^{-01}$		$2.10085 * 10^{-10}$
9	$3.55175 * 10^{-32}$		$1.98477 * 10^{-08}$		$2.95134 * 10^{-10}$

(b) Fixed Size NCL vs. MLP

Inc. Step	Letter	Vehicle	Adult	Optical Digits	Mushroom
1	$3.38568 * 10^{-08}$	$3.18214 * 10^{-04}$	$7.67837 * 10^{-01}$	$1.34057 * 10^{-13}$	$1.32607 * 10^{-01}$
2	$5.78089 * 10^{-09}$	$1.47742 * 10^{-02}$	$4.01307 * 10^{-01}$	$1.72517 * 10^{-14}$	$5.05880 * 10^{-04}$
3	$2.44771 * 10^{-10}$	$1.82929 * 10^{-01}$	$2.62928 * 10^{-02}$	$1.00604 * 10^{-07}$	$1.09935 * 10^{-03}$
4	$9.24056 * 10^{-16}$		$2.84711 * 10^{-01}$	$1.13534 * 10^{-09}$	$1.44943 * 10^{-04}$
5	$5.16720 * 10^{-16}$		$3.29419 * 10^{-01}$	$9.68775 * 10^{-16}$	$2.40132 * 10^{-01}$
6	$2.00037 * 10^{-16}$		$6.51534 * 10^{-01}$	$4.33708 * 10^{-05}$	$7.70052 * 10^{-01}$
7	$3.34724 * 10^{-20}$		$3.88867 * 10^{-01}$		$1.70182 * 10^{-01}$
8	$2.54012 * 10^{-23}$		$8.48938 * 10^{-03}$		$9.99986 * 10^{-01}$
9	$1.18430 * 10^{-23}$		$7.21412 * 10^{-01}$		$7.06443 * 10^{-01}$

Intermediate Summary

In summary, this section shows that the number of ensemble members helps to increase generalisation for both the NCL approaches and SONG. Fixed Size and Growing NCL had comparable generalisation to SONG. Fixed Size NCL was always statistically better than Growing NCL and usually statistically better than a single MLP with the same total number of hidden nodes.

3.1.3.2 Improvement in Generalisation

This section presents an analysis of the improvement in generalisation from the first to the last incremental steps. An improvement occurs when the generalisation in the last incremental step is higher than the generalisation in the first incremental step. A high improvement means that a good incremental ability is attained, while a decrease in the generalisation is a signal of poor incremental learning ability. To support the analysis, the improvement obtained by SONG with 10 SGNTs is also presented.

Table 3.7 shows the average generalisation in the first incremental step, in the last incremental step and the improvement in generalisation. The improvement was calculated as the average generalisation in the last incremental step minus the average generalisation in the first incremental step. In this way, a high number indicates a high improvement. We can define the improvement in generalisation in this way because the chunks are expected to be drawn from the same distribution. Table 3.8 shows the results of the T-Student statistical tests done for the comparison of the improvements.

Table 3.7: Study of Incremental NCL – Improvement in generalisation. The best and the worst improvements for each approach are shown in bold and italic, respectively.

(a) Fixed Size NCL

Database	Generalisation Av.	Improvement
Letter	0.70362-0.81617	**0.11255**
Vehicle	0.77330-0.76898	*-0.00432*
Adult	0.83781-0.84294	0.00513
Optical Digits	0.83509-0.90464	0.06955
Mushroom	0.99233-0.99926	0.00693

(b) Growing NCL

Database	Generalisation Av.	Improvement
Letter	0.66928-0.73193	0.06265
Vehicle	0.76173-0.77269	0.01096
Adult	0.83618-0.83823	*0.00205*
Optical Digits	0.76900-0.90027	**0.13127**
Mushroom	0.99086-0.99377	0.00291

(c) SONG 10 SGNTs

Database	Generalisation Av.	Improvement
Letter	0.81255-0.94323	**0.13068**
Vehicle	0.59691-0.64398	0.04707
Adult	0.79657-0.79924	0.00268
Optical Digits	0.93517-0.96917	0.03400
Mushroom	0.99980-1.00000	*0.00020*

Table 3.8: Study of Incremental NCL – P-values of the T Student Tests for Improvement in Generalisation. For approach A vs. B, the symbols "=", ">" and "<" indicate that A obtained statistically equal, better or worse improvement than B at the level of significance of 0.01.

Database	Fixed Size NCL vs. Growing NCL	Fixed Size NCL vs. SONG 10 SGNTs	Growing NCL vs. SONG 10 SGNTs
Letter	0.00000 >	0.00000 <	0.00000 <
Vehicle	0.01435 =	0.00000 <	0.00000 <
Adult	0.00654 >	0.03895 =	0.49246 =
Optical Digits	0.00000 <	0.00000 >	0.00000 >
Mushroom	0.00015 >	0.00000 >	0.00347 >

We can observe that there is no indication that the improvement is related to the difficulty of the database:

- For Fixed Size NCL, the best improvement occurred for the Letter database and the worst (even negative) for Vehicle.
- For Growing NCL, the best improvement occurred for the Optical Digits database and the worst for Adult.
- For SONG 10 SGNTs, the best improvement occurred for the Letter database and the worst for Mushroom.

We can also observe that, considering T-Student statistical tests with level of significance of 0.01:

- The improvement of Fixed Size NCL was statistically higher than Growing NCL in 3 databases and statistically lower in 1.
- The improvement of Fixed Size NCL was statistically higher than SONG 10 SGNTs in 2 databases and statistically lower in 2.
- The improvement of Growing NCL was statistically higher than SONG 10 SGNTs in 2 databases and statistically lower in 2.

These results show that the improvement for the NCL approaches was similar to the improvement for SONG 10 SGNTs. This observation indicates again that NCL is applicable to incremental learning.

Each approach has at least two databases in which the improvement was very low (less than 0.009). The improvement of Fixed Size NCL for Vehicle was even negative, indicating a poor incremental behaviour of this approach to this database. In order to understand why an approach gets a higher improvement than another and why a specific approach has a higher improvement for some databases than for others, section 3.1.3.3 provides an analysis of the degradation of the accuracy on old training sets as new data sets are used for training.

Intermediate Summary

In summary, both the NCL approaches manage to achieve improvements in generalisation from the first to the last incremental step. The improvement is comparable to SONG.

3.1.3.3 Forgetting

An important analysis to determine whether an approach is forgetting information previously learnt is the analysis of the degradation of the accuracy on old training sets as new data sets

are used for training. To perform this analysis, the training sets previously learnt are used to test the ensemble at each incremental step. It is important to note that they are used only to test, and not to retrain the ensemble.

The degradation of the accuracy for a particular training chunk c_i is defined as follows:

$$d_{c_i} = acc_{t_i}(c_i) - acc_{t_N}(c_i) \tag{3.1}$$

where t_i is the incremental step in which c_i was learnt; t_N is the last incremental step of the learning; and $acc_t(c_i)$ is the accuracy obtained by testing the system using c_i at the end of the incremental step t.

A high decrease in the accuracy from one incremental step to another indicates that there is forgetting. As it will be shown, executions with too much forgetting are likely not to be able to get high improvements in generalisation, explaining the generalisation obtained by the approaches and presented in sections 3.1.3.1 and 3.1.3.2.

The analysis was performed with the Letter and the Vehicle data sets, for which Fixed Size NCL obtained the best and the worst improvement in generalisation, respectively. For both databases, SONG 10 SGNTs obtained a statistically higher improvement in generalisation than the NCL approaches. An analysis was also done using the Optical Digits database, in order to allow comparisons with Learn++ (Polikar et al.; 2001) and Seipone and Bullinaria (2005)'s evolutionary approach. The comparison with these approaches is explained in section 3.1.3.5.

Tables 3.9–3.13 show the average accuracy for each training chunk (rows) in each incremental step (columns). The last column presents the degradation. It is interesting to note that the degradation of the generalisation is equivalent to the opposite of the improvement of the generalisation. However, we will use the term *degradation* to refer only to the degradation of the accuracy of the training chunks.

Fixed Size NCL vs. SONG

When analysing Fixed Size NCL for Letter (table 3.9(a)), for which this approach obtained the highest improvement in generalisation, we can observe that the degradation was not so high (always lower than 0.05). The results also suggest that SONG 10 SGNTs had lower degradations than Fixed Size NCL in most training chunks (5 of 8) for this database (table 3.10), at the same time as it achieved higher improvement in generalisation (section 3.1.3.2).

When analysing Fixed Size NCL for Vehicle (table 3.11(a)), which is the database in which this approach obtained the lowest improvement in generalisation, we can observe that the degradation was very high (always higher than 0.15). The results also suggest that SONG 10 SGNTs had lower degradations than Fixed Size NCL for this database (table 3.12), at the same time as

it achieved higher improvement in generalisation (section 3.1.3.2).

Vehicle vs. Letter vs. Optical Digits

The results suggest that, for Fixed Size NCL, the degradations for the database with the worst improvement in generalisation (Vehicle) were higher than the degradations for the database with the highest improvement in generalisation (Letter). The results also suggest that the database with intermediate improvement in generalisation (Optical Digits, table 3.13(a)) had intermediate degradations. A similar behaviour was presented by SONG 10 SGNTs.

Growing NCL

We can observe that Growing NCL has a very good behaviour in relation to forgetting (tables 3.9(b), 3.11(b) and 3.13(b)), as expected. The degradation is very low and, in many cases, negative. A negative degradation means that the accuracy on the old training chunks is increasing, indicating low forgetting.

However, even with the degradation being very low, Growing NCL's improvement in generalisation was not always better than Fixed Size NCL's (statistically better for Optical Digits, but statistically worse for Letter, Adult and Mushroom), as shown in section 3.1.3.2. Moreover, Growing NCL was always statistically worse than Fixed Size NCL considering the overall generalisation (section 3.1.3.1). The reason for that is that, in Fixed Size NCL, the MLPs used to learn new data already received previous training with data that can have similarities to the new incoming data. So, the initial accuracy of each training chunk is higher (tables 3.9, 3.11 and 3.13). Besides, the higher number of MLPs used by Fixed Size NCL in the beginning of the learning allows it to achieve higher accuracy in the first incremental step.

A possible solution to Growing NCL's problems is to increase the size of the ensemble by more than one MLP for each new data set. The problem of this solution is that the size of the growing ensemble could become very large. Another possible solution is to set the initial weights of the new MLP as the weights of the last previously trained MLP. A third possible solution would be to use not only the new MLP, but also all the MLPs that were already inserted in the ensemble to learn the new data set. This solution could prejudice the Growing NCL's good behaviour to forgetting. These possible solutions are further commented and analysed in section 3.1.4.

Intermediate Summary

The analysis presented in this section suggests a strong relationship between degradation of accuracy and improvement in generalisation for both Fixed Size NCL and SONG 10 SGNTs. It indicates that a high degradation (high forgetting) is related to a low improvement. Even though Growing NCL presents low degradation of accuracy, the fact that only one new MLP is

used to learn each new data chunk reduces its generalisation in comparison to Fixed Size NCL.

Table 3.9: NCL Degradation for Letter.

(a) Fixed Size NCL

Train Chunk \ Step	1	2	3	4	5	6	7	8	9	Degradation
1	0.77339	0.78360	0.79780	0.80185	0.79377	0.79547	0.80845	0.81618	0.81278	-0.03938
2		0.82343	0.79432	0.79547	0.79442	0.79905	0.80948	0.81348	0.81428	0.00915
3			0.84909	0.82361	0.81690	0.82798	0.83403	0.83388	0.83413	0.01495
4				0.84431	0.80020	0.80453	0.80955	0.81643	0.81685	0.02746
5					0.85419	0.81485	0.82481	0.81603	0.81595	0.03823
6						0.86257	0.82908	0.82678	0.82361	0.03896
7							0.85991	0.82646	0.82303	0.03688
8								0.86652	0.83143	0.03508
9									0.86237	0.00000
Generalisation	0.70362	0.77863	0.78808	0.79643	0.79938	0.80467	0.81482	0.81927	0.81617	-0.11255

(b) Growing NCL

Train Chunk \ Step	1	2	3	4	5	6	7	8	9	Degradation
1	0.73256	0.71410	0.73906	0.73743	0.73268	0.73613	0.73808	0.73661	0.73493	-0.00238
2		0.70260	0.72578	0.72396	0.71960	0.72208	0.72656	0.72611	0.72811	-0.02551
3			0.74404	0.74146	0.73718	0.74241	0.74406	0.74229	0.74231	0.00173
4				0.72636	0.72196	0.72666	0.73028	0.73061	0.73031	-0.00395
5					0.73436	0.73798	0.73878	0.73666	0.73633	-0.00198
6						0.73988	0.74239	0.74061	0.73986	0.00003
7							0.73011	0.72871	0.72786	0.00225
8								0.74299	0.74336	-0.00038
9									0.73326	0.00000
Generalisation	0.66928	0.68512	0.71963	0.72197	0.71873	0.72395	0.72997	0.72995	0.73193	-0.06265

Table 3.10: SONG with 10 SGNTs Degradation for Letter.

Train Chunk \ Step	1	2	3	4	5	6	7	8	9	Degradation
1	0.99427	0.96732	0.95712	0.95553	0.95700	0.95853	0.96183	0.96317	0.96212	0.03215
2		0.99955	0.99093	0.98432	0.97940	0.97828	0.97263	0.97003	0.97107	0.02848
3			0.99973	0.99513	0.99235	0.98790	0.98457	0.98470	0.98215	0.01758
4				0.99982	0.99833	0.99387	0.99122	0.98937	0.98698	0.01283
5					0.99988	0.99835	0.99610	0.99483	0.99267	0.00722
6						0.99993	0.99918	0.99810	0.99690	0.00303
7							0.99995	0.99892	0.99820	0.00175
8								0.99993	0.99912	0.00082
9									0.99995	0.00000
Generalisation	0.81255	0.86427	0.89037	0.90720	0.91973	0.92530	0.93165	0.93898	0.94323	-0.13068

Table 3.11: NCL Degradation for Vehicle.

(a) Fixed Size NCL

Train Chunk \ Step	1	2	3	Degradation
1	0.98286	0.86405	0.82167	0.16119
2		0.97024	0.77381	0.19643
3			0.98238	0.00000
Generalisation	0.77330	0.75926	0.76898	0.00432

(b) Growing NCL

Train Chunk \ Step	1	2	3	Degradation
1	0.98071	0.92929	0.88405	0.09667
2		0.80810	0.83262	-0.02452
3			0.79310	0.00000
Generalisation	0.76173	0.72500	0.77269	-0.01096

Table 3.12: SONG with 10 SGNTs Degradation for Vehicle.

Train Chunk \ Step	1	2	3	Degradation
1	0.98651	0.91381	0.86190	0.12460
2		0.99794	0.97651	0.02143
3			0.99984	0.00000
Generalisation	0.59691	0.64089	0.64398	-0.04707

3.1.3.4 Adaptation of Different Ensemble Members

As commented in section 1, one of the advantages of ensembles is that different members can adapt differently to new data. Some of them may have better adaptation, compensating the misclassifications of the others and helping to improve the quality of the ensemble.

This section presents an analysis of the capacity that different neural networks have to adapt differently to new data and the relationship between the similarity among ensemble members and generalisation. The analysis shows that different ensemble members can indeed adapt better or worse for different incremental steps. Even though ensembles can benefit from diverse members, if there is too much diversity, the generalisation can be reduced.

Best Networks of the Ensemble

First, an analysis of the capacity that some MLPs of Fixed Size NCL have to adapt better than the others is presented. This is done by checking what is the neural network of the ensemble which has the best generalisation at each incremental step. In order to do that, a single execution of the 30 Fixed Size NCL executions for each database was chosen. The execution was the one with the intermediate generalisation in the last incremental step. The change in the best MLP from one incremental step to another shows that different MLPs are adapting differently to new

Table 3.13: NCL Degradation for Optical Digits.

(a) Fixed Size NCL

Train Chunk \ Step	1	2	3	4	5	6	Degradation
1	1.00000	0.95940	0.95815	0.93684	0.93659	0.93534	0.06466
2		0.99825	0.96617	0.94912	0.95088	0.90627	0.09198
3			0.99950	0.98070	0.97469	0.96992	0.02957
4				0.99850	0.95363	0.95439	0.04411
5					1.00000	0.96140	0.03860
6						1.00000	0.00000
Generalisation	0.83509	0.89333	0.90543	0.91100	0.92379	0.90464	-0.06955

(b) Growing NCL

Train Chunk \ Step	1	2	3	4	5	6	Degradation
1	0.99248	0.90927	0.92030	0.91429	0.92782	0.91203	0.08045
2		0.87018	0.91504	0.90276	0.93158	0.91353	-0.04336
3			0.94110	0.95013	0.96190	0.95414	-0.01303
4				0.89649	0.91429	0.91830	-0.02180
5					0.93659	0.93484	0.00175
6						0.96817	0.00000
Generalisation	0.76900	0.76743	0.83667	0.87176	0.89698	0.90027	-0.13127

data and some are adapting better than the others.

Table 3.14 shows the best MLP on each incremental step and the percentage of repetition of the best MLP in consecutive incremental steps. Each MLP of the ensemble is identified by a number, from 1 to 10. When more than one MLP has the same best generalisation, all the best MLPs are listed.

Table 3.14: Best MLP of Each Incremental Step.

Incremental Step	Letter Best MLP	Vehicle Best MLP	Adult Best MLP	Opt. Digits Best MLP	Mush. Best MLP
1	2	9	6	5	7
2	7	7	5	4	1
3	7	3	8	4	4, 7
4	8	-	1	4	1, 4, 7
5	8	-	10	4	1
6	7	-	1	2	1, 6
7	3	-	1	-	2, 6, 7, 8
8	2	-	9	-	1, 2, 7
9	4	-	8	-	1, 2, 6, 7
Percentage of Repetition	22% (2/9)	0% (0/3)	11% (1/9)	50% (3/6)	66% (6/9)

We can see that the best MLP of the ensemble from one incremental step to another changes. This shows that, as new data sets come, different MLPs of the ensemble can adapt better. Only for the easiest data sets (Optical Digits and Mushroom) there was a high number

of consecutive incremental steps in which the same MLP appears as the best one, which is a reasonable behaviour. This demonstrates one of the advantages of using an ensemble with diverse neural networks for incremental learning. When a single MLP is utilized to perform incremental learning, it can have a good adaptation to a particular data set, but not to another, decreasing the generalisation of the approach. By using an ensemble, the neural networks that adapt better and faster to new data can compensate the misclassifications of the others. NCL manages to maintain diverse base learners thanks to the negative correlation penalty term, which is explained in section 2.2.

Intersection of the Correct Response Sets

Here, an analysis of the relationship between similarity among ensemble members and generalisation is done for Fixed Size NCL, Growing NCL and SONG 10 SGNTs. The analysis uses the intersection of the correct response sets of the ensemble members for the executions with the intermediate generalisation in the last incremental step. The analysis is intended to present a counter example to show that increasing diversity does not always increases accuracy.

The notion of correct response sets and their intersections was introduced by Liu and Yao (1999b). The correct response set S_i of the individual neural network i on the testing set consists of all patterns in the testing set which are classified correctly by the individual network i. Ω_i denotes the size of the set S_i and $\Omega_{i_1,i_2,...,i_k}$ denotes the size of the set $S_{i_1} \cap S_{i_2} \cap ... \cap S_{i_k}$. In the analysis performed in this section, Ω denotes the percentage of test patterns that is in the intersection, instead of the number of patterns that is in the intersection.

A high Ω indicates that the neural networks participating in the intersection are more similar than a low Ω. It is expected that an ensemble has better generalisation than its individual members when its members are diverse (different from each other) and have accuracy higher than 0.5. So, it is interesting to check whether approaches with the best generalisations are related to lower Ω among all its neural networks.

First, we analyse (1) the intersections between the correct response set of the best and the second best neural networks of each incremental step and (2) the intersection between the best and the worst neural networks of each incremental step, for Fixed Size NCL and SONG 10 SGNTs. This analysis is performed to check the relationship between a high Ω with similar neural networks for the executions analysed. Tables 3.15, 3.16, 3.17, 3.18 and 3.19 show these Ω values for each of the databases.

We can observe that $\Omega_{best,2^{nd}best}$ is usually higher than $\Omega_{best,worst}$, indicating that MLPs or SGNTs with more similar generalisation have also higher Ω. We can also observe that, as new trainings are received by the components of the ensembles, $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$ usually tend to increase. As more training is received by the MLPs or SGNTs, their generalisation become more similar to each other, also indicating that a higher Ω is related to more similar

Table 3.15: Intersection of the Correct Response Sets (Letter) - $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$.

Inc.	Fixed Size NCL		SONG 10 SGNTs	
Step	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$
1	0.60050	0.56650	0.62500	0.61200
2	0.65200	0.62750	0.70600	0.69200
3	0.65600	0.64100	0.73900	0.73300
4	0.68450	0.63150	0.84700	0.76500
5	0.67250	0.63250	0.78900	0.78800
6	0.68050	0.64050	0.80900	0.79900
7	0.66800	0.65000	0.81300	0.80400
8	0.67800	0.65850	0.83100	0.82200
9	0.66000	0.64250	0.83600	0.82100
Av.	0.66133	0.63228	0.77722	0.75956

Table 3.16: Intersection of the Correct Response Sets (Vehicle) - $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$.

Inc.	Fixed Size NCL		SONG 10 SGNTs	
Step	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$
1	0.68056	0.65741	0.44400	0.41700
2	0.68981	0.62037	0.48600	0.47200
3	0.63889	0.62500	0.66200	0.47200
Av.	0.66975	0.63426	0.53067	0.45367

MLPs or SGNTs in the ensemble.

However, it is worth to observe that the increase in $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$ values is not steady. There are some incremental steps in which $\Omega_{best,2^{nd}best}$ and/or $\Omega_{best,worst}$ have considerable decrease in relation to the previous incremental step. Regarding 0.01 as a considerable decrease, Fixed Size NCL has decreases in four out of five databases, while SONG has decrease only in one of the incremental steps of one of the databases. This represents a success of NCL to create and maintain diversity in the ensemble. An extreme case is the Vehicle database, in which the last incremental step has even lower $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$ than the first incremental step when Fixed Size NCL is used.

Now, we analyse the intersection among the correct response sets of all the MLPs/SGNTs of the ensemble Ω_{all} (tables 3.20, 3.21 and 3.22). We can observe that all Ω_{all} values are lower than one, indicating that the neural networks in the ensemble perform differently even being trained with the same data. However, we can also observe that, for the databases in which Fixed Size NCL has higher generalisation than SONG 10 SGNTs (Vehicle and Adult), the intersections of Fixed Size NCL are always higher. For the databases in which SONG 10 SGNTs has higher generalisation than Fixed Size NCL (Letter, Optical Digits and Mushroom), the intersections of SONG are also higher (except the first incremental step of Letter and Mushroom).

This fact shows that higher diversity among ensemble members is not always associated to higher generalisation. In the executions analysed here, a too low Ω_{all} indicates that the

Table 3.17: Intersection of the Correct Response Sets (Adult) - $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$.

Inc.	Fixed Size NCL		SONG 10 SGNTs	
Step	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$
1	0.82224	0.80942	0.67900	0.66600
2	0.80699	0.77294	0.68700	0.68100
3	0.81671	0.77161	0.70100	0.69800
4	0.82268	0.79262	0.69700	0.69400
5	0.83838	0.79239	0.69600	0.69000
6	0.82799	0.80522	0.68800	0.68800
7	0.82467	0.78598	0.69300	0.68400
8	0.82821	0.80544	0.69400	0.68800
9	0.82954	0.80721	0.78400	0.69700
Av.	0.82416	0.79365	0.70211	0.68733

Table 3.18: Intersection of the Correct Response Sets (Optical Digits) - $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$.

Inc.	Fixed Size NCL		SONG 10 SGNTs	
Step	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$
1	0.69887	0.57308	0.88900	0.88200
2	0.79163	0.72715	0.91400	0.90600
3	0.81833	0.75317	0.92600	0.91600
4	0.85611	0.77805	0.93000	0.92200
5	0.87647	0.71154	0.93100	0.92700
6	0.84005	0.72647	0.93900	0.93100
Av.	0.81357	0.71158	0.92150	0.91400

Table 3.19: Intersection of the Correct Response Sets (Mushroom) - $\Omega_{best,2^{nd}best}$ and $\Omega_{best,worst}$.

Inc.	Fixed Size NCL		SONG 10 SGNTs	
Step	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$	$\Omega_{best,2^{nd}best}$	$\Omega_{best,worst}$
1	0.99262	0.98278	1.00000	0.99100
2	0.99631	0.99016	1.00000	0.99800
3	0.99754	0.98647	0.99900	0.99100
4	0.99631	0.98770	0.99900	0.99900
5	0.99877	0.99016	1.00000	1.00000
6	0.99877	0.99139	1.00000	1.00000
7	0.99877	0.99385	1.00000	1.00000
8	0.99877	0.99508	1.00000	1.00000
9	0.99754	0.99508	1.00000	1.00000
Av.	0.99727	0.99030	0.99978	0.99767

components of the ensemble are too different from each other and may reduce the accuracy of the ensemble. Similar results are obtained in chapter 5 when very high levels of diversity are encouraged in a modified version of online bagging. As explained in section 2.1, there are studies in the literature which conclude that high diversity may not always correspond to better generalisation performance.

Table 3.20: Intersection of the Correct Response Sets (Letter and Vehicle) - Ω_{all}.

	Letter		Vehicle	
Incremental Step	Fixed Size NCL Ω_{all}	SONG 10 SGNTs Ω_{all}	Fixed Size NCL Ω_{all}	SONG 10 SGNTs Ω_{all}
1	0.44250	0.34000	0.53704	0.26900
2	0.42900	0.46500	0.42593	0.27800
3	0.46950	0.51600	0.35648	0.23600
4	0.46900	0.55600	-	-
5	0.46950	0.57600	-	-
6	0.48800	0.60300	-	-
7	0.45850	0.61500	-	-
8	0.45500	0.62700	-	-
9	0.45700	0.64100	-	-

Table 3.21: Intersection of the Correct Response Sets (Adult) - Ω_{all}.

	Adult	
Incremental Step	Fixed Size NCL Ω_{all}	SONG 10 SGNTs Ω_{all}
1	0.77824	0.45900
2	0.74066	0.49100
3	0.71391	0.51400
4	0.75702	0.51800
5	0.74707	0.52200
6	0.76476	0.51000
7	0.73049	0.51500
8	0.77382	0.51000
9	0.77625	0.51400

Table 3.23 shows Ω_{all} for Growing NCL. We can observe that Ω_{all} always decreases from one incremental step to another. This is a reasonable behaviour, as a new MLP is added to the ensemble at each incremental step. Apart from the second and third incremental steps, Growing NCL's Ω_{all} is always lower than Fixed Size NCL's. Again, a relation between a too low Ω_{all} and worse generalisation is indicated.

Intermediate Summary

The analysis shows through an example that different ensemble members are indeed able

Table 3.22: Intersection of the Correct Response Sets (Optical Digits and Mushroom) - Ω_{all}.

	Optical Digits		Mushroom	
Incremental Step	Fixed Size NCL Ω_{all}	SONG 10 SGNTs Ω_{all}	Fixed Size NCL Ω_{all}	SONG 10 SGNTs Ω_{all}
1	0.37262	0.79400	0.44250	0.34000
2	0.53846	0.82500	0.42900	0.46500
3	0.52670	0.84200	0.46950	0.51600
4	0.54751	0.85100	0.46900	0.55600
5	0.66290	0.85500	0.46950	0.57600
6	0.69434	0.85600	0.48800	0.60300
7	-	-	0.45850	0.61500
8	-	-	0.45500	0.62700
9	-	-	0.45700	0.64100

Table 3.23: Intersection of the All Correct Response Sets For Growing NCL - Ω_{all}. The first incremental step listed in the table is the second incremental step, as there is only one MLP in the first incremental step.

Incremental Step	Letter	Vehicle	Adult	Opt. Digits	Mush.
2	0.58900	0.57870	0.75039	0.67964	0.98401
3	0.48650	0.46759	0.73447	0.55814	0.93850
4	0.36350	-	0.72761	0.46176	0.92497
5	0.28950	-	0.71656	0.43484	0.92497
6	0.13600	-	0.69246	0.37127	0.83026
7	0.12900	-	0.67698	-	0.83026
8	0.12850	-	0.66947	-	0.83026
9	0.11250	-	0.62945	-	0.78106

to adapt better or worse to new data and that NCL manages to maintain diverse members as new training chunks are received. Even though ensembles can benefit from diverse members, if there is too much diversity, the generalisation can be reduced.

3.1.3.5 Comparison to Additional Approaches

This section presents a comparison between the NCL incremental approaches with Learn++ and Seipone and Bullinaria (2005)'s evolutionary approach. The comparison uses the results published in the literature. So, it is not possible to use statistical tests.

Tables 3.24 and 3.25 show the degradation and generalisation for Learn++ and the evolutionary approach. When comparing them with tables 3.11 and 3.13, we can observe that the NCL approaches usually get lower generalisation. It is important to remember that the evolutionary approach considers all the data sets in each generation, being biased, and that Learn++ used ensembles with 30 members, whereas the NCL approaches used at most size 10. This may influence the generalisation.

The positive point of the NCL approaches is regarding degradation and improvement in generalisation. Growing NCL managed to get lower or comparable degradation to Learn++ and usually lower degradation than the evolutionary approach. Growing NCL also got higher improvement in generalisation than Learn++ for Optical Digits and higher improvement than the evolutionary approach. Fixed Size NCL also managed to get higher generalisation improvement than the evolutionary approach. So, an approach which combines the Growing NCL robustness against forgetting to the use of a whole ensemble to learn new data may achieve higher generalisation.

Table 3.24: Learn++ Degradation for Vehicle.

Train Set \ Inc	1	2	3	Degradation
1	0.930	0.820	0.790	0.140
2		0.860	0.780	0.080
3			0.100	0.000
Generalisation	0.780	0.804	0.83000	-0.05

Table 3.25: Learn++ and Evolutionary Approach Degradation for Optical Digits.

(a) Learn++

Train Set \ Inc	1	2	3	4	5	6	Degradation
1	0.940	0.940	0.940	0.930	0.930	0.930	0.010
2		0.935	0.940	0.940	0.940	0.930	0.005
3			0.950	0.940	0.940	0.940	0.010
4				0.935	0.940	0.940	-0.005
5					0.950	0.950	0.000
6						0.950	0.000
Generalisation	0.820	0.847	0.897	0.917	0.922	0.927	-0.107

(b) Evolutionary Approach

Train Set \ Inc	1	2	3	4	5	6	Degradation
1	1.000	0.988	0.984	0.981	0.979	0.980	0.020
2		1.000	0.989	0.983	0.980	0.978	0.022
3			1.000	0.990	0.985	0.983	0.017
4				1.000	0.991	0.985	0.015
5					1.000	0.989	0.011
6						1.000	0.000
Generalisation	0.914	0.929	0.936	0.939	0.942	0.944	-0.030

3.1.4 Selective NCL

As explained in section 3.1.3, Growing NCL is able to reduce forgetting, but has low generalisation. As commented in that section, the following possible solutions are suggested to help overcoming Growing NCL's problems:

- Increase the size of the ensemble by more than one MLP to each new data chunk. The

problem is that the size of the growing ensemble could become very large.

- Set the initial weights of the new MLP as the weights of the last previously trained MLP.
- Use not only the new MLP, but also all the MLPs that were already inserted in the ensemble to learn the new data set. This solution could prejudice Growing NCL's good behaviour against forgetting.

Ke Tang proposed to use all these three solutions at the same time, by combining NCL with a selection procedure, producing a work in collaboration (Tang et al.; 2009). In Selective NCL, whenever a new data chunk is received, the previously trained ensemble is cloned. The clone is trained on the new chunk and then combined with the previous ensemble. A selection process is then applied to prune the whole ensemble to a fixed size, eliminating the problem of a too large ensemble after a long period of training.

Experiments using Letter, Vehicle and Optical Digits with the setup explained in section 3.1.2 and two additional databases were done. A selection procedure based on Genetic Algorithms (GAs) (De Jong; 2006) and a repair operator to eliminate infeasible solutions was adopted.

ANOVA and multiple comparison tests (Montgomery; 2004) were used to show that Selective NCL suffered more from forgetting than Growing NCL, but less than Fixed Size NCL, depending on the selection procedure setup. The approach achieved statistically higher generalisation than Growing NCL for two databases and statistically similar generalisation for the others. It achieved statistically higher generalisation than Fixed Size NCL for two databases, statistically worse for one and statistically equal for the others. So, it reduced forgetting in relation to Fixed Size NCL at the same time as it usually increased or maintained generalisation in relation to Growing NCL and Fixed Size NCL.

This behaviour makes Selective NCL more competitive against Learn++. Considering the generalisation obtained in the last incremental step and the Learn++ results published in the literature (Polikar et al.; 2001), Selective NCL is better than Learn++ for Optical Digits and worse for Vehicle. One of its main advantages over Learn++ is the fact that it has a fixed ensemble size, whereas Learn++ grows indefinitely as new data chunks are made available.

The disadvantage of Selective NCL is that its training time is considerably higher than the other NCL approaches, due to the GA-based selection procedure adopted.

3.1.5 Discussion

The analysis shows that it is possible to use NCL in incremental learning, although each approach also has its weaknesses. Fixed Size NCL uses several different neural networks to learn new data, getting advantage from the fact that different networks adapt differently to new data and

achieving higher generalisation than Growing NCL. Both Fixed Size and Growing NCL have comparable generalisation to SONG, which is an approach existing in the literature. Growing NCL manages to overcome forgetting, but the fact that it uses only one new neural network to learn new data affects its generalisation badly. Besides, Growing NCL can increase its size indefinitely as new training chunks arrive.

A relationship between low/high degradation (forgetting) and high/low improvement in generalisation is suggested for both Fixed Size NCL and SONG. Even though Growing NCL has a good behaviour against forgetting, the use of only one new neural network to learn new data did not not allow it to outperform other approaches in terms of generalisation and a lower degradation was not always associated to a higher improvement in generalisation.

Both the approaches managed to get improvements in the generalisation from the first to the last incremental step. The improvement was comparable to other approaches specifically developed to incremental learning existent in the literature. This result is very encouraging, indicating that new approaches combining Fixed Size and Growing NCL may overcome the improvements achieved by the current approaches and obtain even better generalisation.

Selective NCL was able to reduce forgetting in comparison to Fixed Size NCL and obtained usually similar or higher generalisation than Growing NCL and Fixed Size NCL. However, its training time is considerably higher and further investigation is still necessary in order to achieve higher, instead of comparable generalisation to other approaches such as Learn++. Further investigation to reduce Selective NCL's training time by adopting a faster selection procedure and the analysis of the NCL approaches to the presentation of new classes in the incoming data are also proposed as future work.

This section also illustrates one of the problems of the existing incremental learning approaches designed for dealing with changing environments. As explained in section 2, most of these approaches give little attention to the stability of the classifiers, giving more emphasis to the plasticity, as they usually allow only a new classifier to learn a new data chunk. When there are very frequent concept drifts, this feature is desirable because it makes the system very plastic. However, when there are long periods of stability or when the drifts are less frequent than the chunk size, the fact that at most one ensemble member learns each new training example does not allow the system to achieve as high accuracy as if each training example contributed to the learning of several ensemble members.

We can observe this behaviour when comparing Fixed Size NCL with Growing NCL. The difference between these two approaches is the fact that the former allows the whole ensemble to learn new data, whereas the latter allows only a new ensemble member to learn new data. Because of that, Fixed Size NCL achieves higher accuracy than Growing NCL when using data sets which contain no (known) drift. If the training chunks were very large and contained mostly redundant information in comparison to each other, Growing NCL might have its accuracy improved, as this would mimic the behaviour of a fixed size approach after the ensemble achieved

a certain size. However, this setup is desirable neither for stable nor for changing environments. Independent of the environment, the time and memory requirements would be increased due to the large chunks. Considering changing environments, a large chunk size would not allow dealing properly with concept drift, as explained in chapter 2. This demonstrates the problem presented by most of the incremental approaches designed for dealing with concept drift in the literature.

3.2 Negative Correlation Approaches for Online Learning

This section presents a study of online NCL. Two different approaches (section 3.2.1) are analysed: direct application of NCL in online environments (which we will call simply NCL) and online bagging NCL. The former approach sends the same sequence of training examples to all base learners. So, the choice of base models is restricted to neural networks that become different from each other even when trained with the same sequence of data, such as MLPs. The latter approach sends a different sequence of training examples, allowing a wider range of choice for the base model. A third approach, online clustering NCL, is also commented.

The analysis, which is given in section 3.2.3 and whose design is explained in section 3.2.2, performs the following comparisons:

- Comparison between NCL using MLPs in online and offline mode, in order to check whether the accuracy is too much reduced when using MLPs in online mode (section 3.2.3.1).
- Comparison between NCL and online bagging NCL, to validate online bagging NCL and check whether its accuracy can be improved in comparison to NCL through the use of more appropriate base learners (section 3.2.3.2).

The analysis shows that NCL using online MLPs has (sometimes drastically) higher (worse) classification error than using offline MLPs in four out of five classification databases used in the study. The only database in which it achieves similar error is a large database. Online bagging NCL using Evolving Fuzzy Neural Networks (EFuNNs) is able to reduce the error in four out of five databases thanks to the use of EFuNNs as the base learners. It even manages to achieve similar generalisation error to NCL using offline MLPs. This is a very good result, as the EFuNNs use each training example only once, while offline MLPs can use the whole training set a certain number of epochs. The analysis shows the importance of a wider range of choice for the base learner, which is allowed by online bagging NCL.

3.2.1 Direct Application of NCL and Online Bagging NCL

One of the ways to perform online NCL is to directly use NCL, as explained in section 2.2, with online neural networks as the base learners. The problem of this approach is that, even though

it is parallel-generation multiple-update, it sends the same sequence of training examples to all the base learners. So, it can only be used with base learners which become different from each other even when the same sequence of data is learnt. In order to overcome this problem, a new approach called online bagging NCL, which sends different sequences of training examples for different base learners, is proposed.

Similarly to online bagging, online bagging NCL presents each training example W times to the online learning algorithm, where W is drawn from a $Poisson(1)$ distribution. The difference is that the base learners use an error function with NCL's negative correlation penalty term. So, before the learning of a training example d, the average of the outputs of the base learners on d is calculated to be used by their learning algorithm. Algorithm 3.3 outlines online bagging NCL.

Algorithm 3.3 Online Bagging NCL

Inputs: ensemble **h**, ensemble size M, training example d, strength parameter γ and online learning algorithm $OnlineBaseLearningAlg$ which uses an error function adapted to the use of the NCL penalty term.

1: Calculate the ensemble output for the training example d, $F(d)$.
2: **for** $m \leftarrow 1$ to M **do**
3: $W \leftarrow Poisson(1)$
4: **for** $w \leftarrow 1$ to W **do**
5: $h_m \leftarrow OnlineBaseLearningAlg(h_m, F(d), \gamma, d)$
6: **end for**
7: **end for**

Output: updated ensemble **h**.

3.2.2 Experimental Design

The objective of the experiments is to analyse the points explained in the beginning of section 3.2. The databases used in the experiments were Adult, Letter Recognition, Mushroom, Optical Recognition of Handwritten Digits and Vehicle Silhouettes, from the UCI Machine Learning Repository (Newman et al.; 2010), as in section 3.1. The number of input attributes, classes and examples of each database are shown in table 3.26.

Table 3.26: Databases for the Study of Online NCL.

Database	Inputs	Classes	Examples
Adult	14	2	45222
Letter	16	26	20000
Mushroom	21	2	8124
Optdigits	64	10	5620
Vehicle	18	4	846

Two base learners were used: online MLPs trained with stochastic backpropagation and

Evolving Fuzzy Neural Networks (EFuNNs). MLPs are the neural networks most frequently used in the studies of NCL. EFuNNs (Kasabov; 2001) are local and constructive online neural networks and are briefly explained in appendix A.

The parameters used in the experiments, except the learning rate used for the stochastic back-propagation, were chosen after visual inspection of preliminary executions varying the parameters. The number of hidden nodes used for both online and offline MLPs and the number of epochs and the learning rate used for the offline MLPs are shown in table 3.27.

For the online stochastic back-propagation, five runs of 2-fold cross-validation were performed for each of the six learning rates in $\{1, 0.1, 0.01, ..., 0.00001\}$, to guarantee that the classification errors obtained are not resulting from a bad learning rate choice. The best classification error averages were attained by using learning rate 0.1 for all databases but Vehicle, in which the best classification error average was obtained by using 0.01. The results reported in the rest of the study are the ones obtained with the best learning rates for each database.

Experiments using five runs of 2-fold cross-validation were also done presenting each training example a certain number of times (more than 1) "on-arrival" and then discarded, as it was done by Oza and Russell (2005) to perform online learning. However, the accuracies did not improve in comparison to the use of offline MLPs. So, the rest of section 3.2 shows only the results obtained by presenting each training example only once "on-arrival".

Table 3.27: Offline MLP Parameters for the Study of Online NCL. The number of hidden nodes was also used for online MLPs.

Database	Hidden nodes	Epochs	Learning rate
Adult	5	100	0.1
Letter	40	300	0.1
Mushroom	25	100	0.05
Optdigits	10	100	0.1
Vehicle	30	1500	0.1

The EFuNN parameters were the following: error threshold $E = 0.1$, initial sensibility threshold $S = 0.9$, maximum radius of receptive field $Mrad = 0.5$, membership functions number = 3, membership functions type = triangular, m-of-n = 3, no rules extraction, no aggregation and no pruning, except for Adult, in which pruning was used with $Pr = 1$ and node age $OLD = 200$, and Vehicle, in which the error threshold was $E = 0.001$.

The ensembles created in the experiments were composed of 10 base learners combined by majority-vote and the penalty strength was $\gamma = 0.4$.

Five runs of 2-fold cross-validation were performed with the chosen parameters. All the comparisons presented in sections 3.2.3.1 and 3.2.3.2 consider the classification errors obtained after the presentation of all available training examples and are confirmed by 5x2 cross-validation

F tests with 95% of confidence, as recommended by Dietterich (1998) and Alpaydin (1999). When the difference between two averages is statistically significant and one of the averages is better/worse than the other, we will simply refer to it as being statistically better/worse than the other, to simplify the writing.

3.2.3 Analysis

3.2.3.1 Online NCL vs. Offline NCL

This section presents a comparison between NCL using online and offline stochastic back-propagation MLPs. It indicates that NCL with online MLPs is unsuitable for online learning, due to the high classification errors obtained when the databases are not large.

The classification error averages of the ensembles of online MLPs and offline MLPs produced by NCL are shown in figure 3.2. We can observe that both the train and test (generalisation) errors obtained by the ensembles of online MLPs were usually considerably and sometimes even drastically increased in comparison with the use offline MLPs. Except for the Adult database, the classification errors obtained using online MLPs were always more than twice the classification errors obtained using offline MLPs. For Letter and Vehicle, the classification error averages using online MLPs are very high.

For all databases but Adult, the statistical tests indicate that NCL with online MLPs produced worse classification error than NCL with offline MLPs. Table 3.28 shows the averages, standard deviations and f statistic of the 5x2 cross-validation F tests. The f statistics higher than 4.74 indicate that there is a statistically significant difference between the averages with 95% of confidence. These statistics are marked with the symbol "*" in this and all the other tables of section 3.2.

The Adult database has a large number of training examples. This may be the reason why the classification errors of NCL with online and offline MLPs were statistically the same. That behaviour agrees with the intuition. When the database is large and has no drifts, several examples may be similar to each other, having the same effect as the use of several epochs in the offline learning of a smaller database.

The experiments presented in this section indicate that NCL using online MLPs is not suitable to online learning when the database is not large. As the only difference between NCL applied in online and offline mode is the base model, it is likely that the bad classification errors obtained are due to the online MLPs. Online bagging NCL allows the use of more suitable base models. Section 3.2.3.2 presents experiments with this approach, confirming that online MLPs are not so suitable for online learning when the databases are not large as other classifiers such as EFuNN. So, the possibility of a wider range of choice for base model is an important characteristic of online bagging NCL.

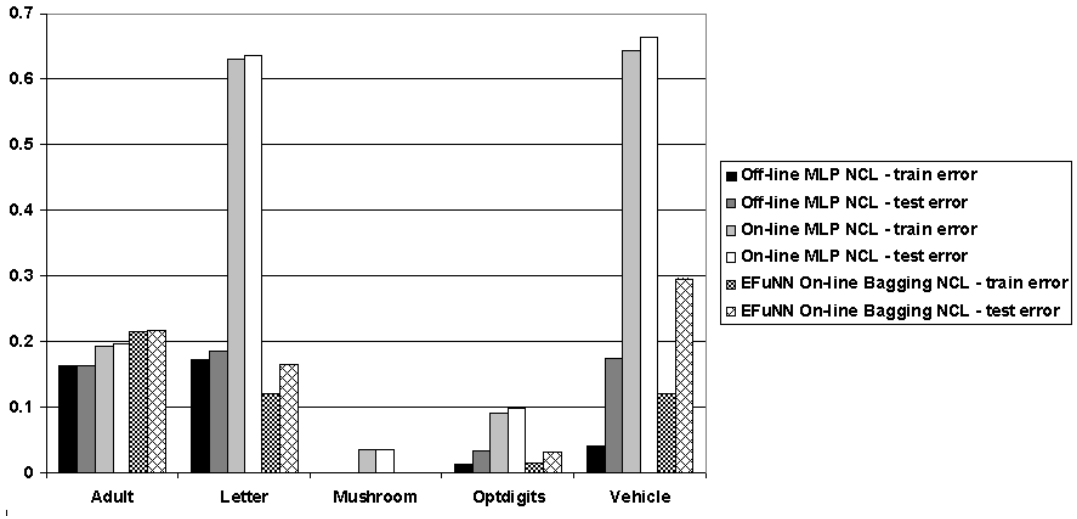


Figure 3.2: Study of Online NCL – Average classification error. The average train and test error using NCL with offline MLPs for Mushroom were very small (0.00000 and 0.00020, respectively), not appearing in the graph.

Table 3.28: Classification error averages (Av), standard deviations (SD) and f statistics of the 5x2 cross-validation F tests (Alpaydin; 1999) of the ensembles of online MLPs and offline MLPs trained with NCL. The f values marked with the symbol "*" indicate a statistically significant difference with 95% confidence. The f values are truncated to facilitate visualisation. None of the truncations affect the results of the comparison against 4.74 to determine whether there is statistically significant difference.

	Train error			Test error		
	NCL with Online MLPs Av +- SD	NCL with Offline MLPs Av +- SD	f	NCL with Online MLPs Av +- SD	NCL with Offline Av +- SD	f
Adult	0.19383 +- 0.01980	0.16193 +- 0.01438	3	0.19633 +- 0.01848	0.16247 +- 0.01222	3
Letter	0.63195 +- 0.04651	0.17184 +- 0.00693	*97	0.63435 +- 0.04768	0.18577 +- 0.00863	*77
Mush.	0.03508 +- 0.00418	0.00000 +- 0.00000	*56	0.03543 +- 0.00495	0.00020 +- 0.00062	*46
Opt.	0.09135 +- 0.01967	0.01331 +- 0.00170	*12	0.09868 +- 0.02200	0.03274 +- 0.00428	*9
Vehicle	0.64255 +- 0.04712	0.04043 +- 0.00682	*614	0.66478 +- 0.05651	0.17494 +- 0.02654	*111

3.2.3.2 Online NCL vs. Online Bagging NCL

The classification error averages of the ensembles of EFuNNs produced by online bagging NCL and of the ensembles of online MLPs produced by NCL are shown in figure 3.2. Table 3.29 shows the classification error averages, standard deviations and f statistics of the 5x2 cross-validation F tests done to compare these approaches. The experiments show that for all databases but Adult, online bagging NCL with EFuNNs obtains a statistically better (lower) classification error than NCL with online MLPs. Again, the classification errors for Adult, which is a large database, were statistically the same.

These results show that online bagging using EFuNNs can get better classification error than NCL using online MLPs, validating the approach. However, it is important to check whether online bagging NCL with EFuNNs outperforms NCL with online MLPs because of the possibility to choose EFuNNs as the base models or because of online bagging. In order to do that, the following two comparisons were done:

1. NCL with online MLPs versus online bagging NCL with online MLPs - to check whether online bagging NCL by itself is improving or not the classification error in relation to NCL.
2. Online bagging NCL with online MLPs versus online bagging NCL with EFuNNs - to complement the first comparison, checking the importance of the base learner to the performance of online bagging NCL.

Table 3.30 shows the classification error averages, standard deviations and f statistics of the 5x2 cross-validation F tests for the first comparison. It is possible to observe that online bagging NCL with online MLPs obtained either statistically equal or worse classification error averages than NCL with online MLPs. So, online bagging NCL by itself could not improve the classification in relation to NCL.

Table 3.31 shows the results for the second comparison. Online bagging NCL with EFuNNs obtained a statistically better classification error than online bagging NCL with online MLPs for all databases but Adult.

These two comparisons show that EFuNNs played a very important role in improving online bagging NCL's classification. So, the possibility to choose deterministic classifiers such as EFuNNs is an important characteristic of online bagging NCL, making it possible to improve its generalisation in relation to NCL for smaller databases.

Table 3.29: Classification error averages (Av), standard deviations (SD) and f statistics of the 5x2 cross-validation F tests (Alpaydin; 1999) of the ensembles of online MLPs trained with NCL and of the ensembles of EFuNNs trained with online bagging NCL. The f values marked with the symbol "*" indicate a statistically significant difference with 95% confidence. The f values are truncated to facilitate visualisation. None of the truncations affect the results of the comparison against 4.74 to determine whether there is statistically significant difference.

	Train error			Test error		
	NCL with online MLPs Av +- SD	Online bagging NCL with EFuNNs Av +- SD	f	NCL with online MLPs Av +- SD	Online bagging NCL with EFuNNs Av +- SD	f
Adult	0.19383 +- 0.01980	0.21567 +- 0.01858	0	0.19633 +- 0.01848	0.21852 +- 0.01963	0
Letter	0.63195 +- 0.04651	0.12192 +- 0.00457	*120	0.63435 +- 0.04768	0.16530 +- 0.00357	*85
Mush.	0.03508 +- 0.00418	0.00007 +- 0.00012	*55	0.03543 +- 0.00495	0.00010 +- 0.00013	*49
Opt.	0.09135 +- 0.01967	0.01630 +- 0.00157	*11	0.09868 +- 0.02200	0.03128 +- 0.00395	*8
Vehicle	0.64255 +- 0.04712	0.12222 +- 0.00965	*1365	0.66478 +- 0.05651	0.29551 +- 0.01142	*231

Table 3.30: Classification error averages (Av), standard deviations (SD) and f statistics of the 5x2 cross-validation F tests (Alpaydin; 1999) of the ensembles of online MLPs trained with NCL and of the ensembles of online MLPs trained with online bagging NCL. The f values marked with the symbol "*" indicate a statistically significant difference with 95% confidence. The f values are truncated to facilitate visualisation. None of the truncations affect the results of the comparison against 4.74 to determine whether there is statistically significant difference.

	Train error average			Test error average		
	NCL with Online MLP Av +- SD	Online Bagging NCL with Online MLP Av +- SD	f	NCL with Online MLP Av +- SD	Online Bagging NCL with Online MLP Av +- SD	f
Adult	0.19383 +- 0.01980	0.21034 +- 0.04975	1	0.19633 +- 0.01848	0.21266 +- 0.05008	1
Letter	0.63195 +- 0.04651	0.76792 +- 0.05509	3	0.63435 +- 0.04768	0.77045 +- 0.05412	3
Mush.	0.03508 +- 0.00418	0.06278 +- 0.01293	*7	0.03543 +- 0.00495	0.06381 +- 0.01530	*7
Opt.	0.09135 +- 0.01967	0.17651 +- 0.02607	*6	0.09868 +- 0.02200	0.18552 +- 0.02737	*6
Vehicle	0.64255 +- 0.04712	0.67021 +- 0.04291	2	0.66478 +- 0.05651	0.68156 +- 0.04213	0

Table 3.31: Classification error averages (Av), standard deviations (SD) and f statistics of the 5x2 cross-validation F tests (Alpaydin; 1999) of the ensembles of online MLPs trained with online bagging NCL and of the ensembles of EFuNNs trained with online bagging NCL. The f values marked with the symbol "*" indicate a statistically significant difference with 95% confidence. The f values are truncated to facilitate visualisation. None of the truncations affect the results of the comparison against 4.74 to determine whether there is statistically significant difference.

	Train error average			Test error average		
	Online Bagging NCL		f	Online Bagging NCL		f
	Online MLP Av +- SD	EFuNN Av +- SD		Online MLP Av +- SD	EFuNN Av +- SD	
Adult	0.21034 +- 0.04975	0.21567 +- 0.01858	0	0.21266 +- 0.05008	0.21852 +- 0.01963	1
Letter	0.76792 +- 0.05509	0.12192 +- 0.00457	*103	0.77045 +- 0.05412	0.16530 +- 0.00357	*95
Mush.	0.06278 +- 0.01293	0.00007 +- 0.00012	*41	0.06381 +- 0.01530	0.00010 +- 0.00013	*28
Opt.	0.17651 +- 0.02607	0.01630 +- 0.00157	*30	0.18552 +- 0.02737	0.03128 +- 0.00395	*26
Vehicle	0.67021 +- 0.04291	0.12222 +- 0.00965	*190	0.68156 +- 0.04213	0.29551 +- 0.01142	*71

Table 3.32: Classification error averages (Av), standard deviations (SD) and f statistics of the 5x2 cross-validation F tests (Alpaydin; 1999) of the ensembles of offline MLPs trained with NCL and of the ensembles of EFuNNs trained with online bagging NCL. The f values marked with the symbol "*" indicate a statistically significant difference with 95% confidence. The f values are truncated to facilitate visualisation. None of the truncations affect the results of the comparison against 4.74 to determine whether there is statistically significant difference.

	Train error			Test error		
	NCL with offline MLPs Av +- SD	Online bagging NCL with EFuNNs Av +- SD	f	NCL with offline MLPs Av +- SD	Online bagging NCL with EFuNNs Av +- SD	f
Adult	0.16193 +- 0.01438	0.21567 +- 0.01858	3	0.16247 +- 0.01222	0.21852 +- 0.01963	4
Letter	0.17184 +- 0.00693	0.12192 +- 0.00457	*38	0.18577 +- 0.00863	0.16530 +- 0.00357	3
Mush.	0.00000 +- 0.00000	0.00007 +- 0.00012	1	0.00020 +- 0.00062	0.00010 +- 0.00013	1
Opt.	0.01331 +- 0.00170	0.01630 +- 0.00157	*6	0.03274 +- 0.00428	0.03128 +- 0.00395	2
Vehicle	0.04043 +- 0.00682	0.12222 +- 0.00965	*26	0.17494 +- 0.02654	0.29551 +- 0.01142	*26

The comparison between online bagging NCL with EFuNNs and NCL with offline MLPs is also important to show the benefits of online bagging NCL. Figure 3.2 shows the classification error averages obtained by both the approaches and table 3.32 shows the results of the comparisons. There is no statistically significant difference between the test classification errors obtained by online bagging NCL with EFuNNs and NCL with offline MLPs, except for the Vehicle database, which is a small database. This is an impressive result, as in offline learning the MLPs can process the whole training set a certain number of epochs, while EFuNNs process each training example only once.

3.2.4 Online Clustering NCL

Another way to send a different sequence of training examples to each base learner is by using a growing approach based on an online clustering method. We will refer to this approach as Online Clustering NCL. In this approach, a neural network can be associated to each cluster. When a new cluster is created, a new neural network is introduced into the system. So, different base models receive different sequences of data because they are created in different moments. Algorithm 3.4 outlines the approach.

Algorithm 3.4 Online Clustering NCL

Inputs: ensemble $\mathbf{h}$, clusters $\mathbf{c}$, ensemble size M, training example d, strength parameter γ, online learning algorithm $OnlineBaseLearningAlg$ which uses an error function adapted to the use of the NCL penalty term, and online clustering algorithm $OnlineClusteringAlg$.

1: Calculate $F(d)$.
2: **for** $m \leftarrow 1$ to M **do**
3: $K \leftarrow Poisson(1)$
4: **for** $k \leftarrow 1$ to K **do**
5: $h_m \leftarrow OnlineBaseLearningAlg(h_m, F(d), \gamma, d)$
6: **end for**
7: **end for**
8: **if** $\mathbf{c} = OnlineClusteringAlg(\mathbf{c}, d)$ creates a new cluster **then**
9: $h_{M+1} = OnlineBaseLearningAlg(h_{M+1}, d)$
10: $M \leftarrow M + 1$
11: **end if**

Output: updated ensemble $\mathbf{h}$, clusters $\mathbf{c}$ and ensemble size M.

Even though all the existing neural networks are potentially trained with the incoming training examples, the ensemble has a small size during the beginning of the learning. So, this approach cannot use the full power of ensembles to achieve higher accuracy while the size is too small.

Experiments to compare online bagging NCL using EFuNNs with online clustering NCL using EFuNNs were done using the same setup as in section 3.2.2. The online clustering algorithm was the Evolving Clustering Method (Kasabov and Song; 2002) and its threshold for creating a new cluster was chosen so as to generate in total around ten neural networks. The analysis

revealed that online clustering NCL got worse test classification error in three databases (Adult, Letter and Optical Digits). The difference in test classification error was considered statistically significant at the 5% level using 5x2 cross-validation F tests. In the remaining databases (Mushroom and Vehicle), both approaches performed similarly (statistically equal classification errors).

When comparing the execution time between the two approaches, online clustering NCL obtained lower training time for Adult and Optical Digits. The difference in training time was considered statistically significant at the 5% level using 5x2 cross-validation F tests. For the other databases, there was no statistically significant difference between the training times. As online clustering NCL is a growing approach, it has less ensemble members for a certain time period, achieving faster training. However, there is no guarantee that the total training time will be always faster. If many clusters are created during the lifelong learning, online clustering NCL may start requiring longer training time.

Online clustering NCL might be able to achieve better generalisation once its size becomes large enough. However, there would always be the problem of starting with a small size, which is likely to cause worse generalisation for a certain period of time.

3.2.5 Discussion

This section presents a study of online NCL. The direct application of NCL in online learning sends the same sequence of training examples for all the base learners. So, even though it has the strong point of being a parallel-generation multiple-update approach, the choice of base models is restricted to neural networks that become different from each other even when trained with the same sequence of data, such as MLPs. In the experiments performed in this section, NCL using online MLPs got high classification error especially when the databases were not large.

A new approach called online bagging NCL is proposed in order to send a different sequence of training examples to each base learner. In this way, a wider range of base models can be used, including more adequate models, such as EFuNNs. In online bagging NCL, the training of an ensemble member is influenced by the training of the others, directly encouraging diversity. This is an advantage of this approach over the other online ensemble approaches existent in the literature, except NCL.

The experiments show that online bagging NCL using EFuNNs outperformed NCL using online MLPs in 4 out of 5 databases. The difference in classification error for these databases was statistically significant at the level 5% using 5x2 cross-validation F tests. These results show the importance of a wider range of choice for the base learners. Online bagging NCL allowed the use of EFuNNs as the base model, which performs local and constructive learning. Local and constructive learning allows the neural network model to start initially very small and grow as more training examples are used, as shown in algorithm A.6, appendix A. This is advantageous

especially when there are few training examples, as it is difficult to learn large models based on few data.

Online bagging NCL using EFuNNs managed to attain similar (statistically equal) test classification error to NCL using offline MLPs in 4 out of 5 databases. This is a very good result, as in offline learning the MLPs can process the whole training set a certain number of epochs, while EFuNNs process each training example only once. This analysis emphasizes even more the importance of having a wider range of choices of base model, which is allowed by online bagging NCL.

Another approach, called online clustering NCL, can also be used to send a different sequence of training examples for each ensemble member. However, this approach is likely to get worse generalisation while the ensemble contains few members.

3.3 Summary

The main contribution of this chapter is a study of NCL in incremental and online learning. In incremental learning, the study shows that NCL is promising, even though each of the approaches analysed has its weaknesses. Fixed Size NCL can benefit from diversity among base learners trained with all the incoming data, achieving better generalisation than Growing NCL. Growing NCL is more successful in avoiding forgetting. However, it uses only one neural network to learn new data, getting worse generalisation. Both the approaches manage to get improvements in the generalisation from the first to the last incremental step, showing that they can learn useful information from new data. This result is very encouraging, showing that an approach combining the advantages of Growing and Fixed Size NCL may achieve even better generalisation. Another approach, called Selective NCL, manages to achieve usually similar or better generalisation to the former approaches, but additional investigation is still necessary in order to further improve the generalisation in comparison to other approaches such as Learn++.

The study of NCL in incremental learning also illustrates one of the problems of the existing incremental approaches designed for dealing with changing environments. Those approaches usually generate a new classifier to learn each new data chunk. In this way, they cannot benefit from the use of several classifiers to learn each training example and are likely to get lower accuracy during periods of stable concept.

The study of NCL in online learning introduces a new approach, online bagging NCL, which besides being parallel-generation multiple-update, it sends a different sequence of training examples to the base learners. In this way, it allows a wider range of choices for the base learners. Thanks to that, it was able to outperform the direct application of NCL to online learning when the databases were not large. Besides the direct application of NCL to online learning and online bagging NCL, an approach called clustering NCL was also proposed. However, it achieved similar

or worse accuracy than online bagging NCL.

This chapter also suggests not to use NCL for the study of the impact of diversity in online learning in the presence of concept drift, which is necessary for answering research question 1.2.2. The reason is that online bagging NCL, which is likely to be more accurate, uses more than one source of diversity: from NCL's penalty term and from online bagging's poisson distribution. An approach with only one source of diversity which can be directly controlled to consistently generate more or less diversity is more desirable for performing a principled study. An online ensemble learning approach which allows consistently "tuning" diversity is introduced in chapter 5. Chapter 4 explains important points related to conducting principled studies of concept drift in general.

Chapter 4

Preparing Principled Studies of Concept Drift

When performing studies of drifts, it is important to determine how existing and new approaches or strategies behave considering different types of drift. So, in order to perform principled studies, it is necessary to use a clear categorisation, separating drifts according to different criteria into mutually exclusive and non-heterogeneous categories. It is also necessary to use data sets with known drifts which represent different categories. Section 4.1 explains why the current categories existing in the literature are not appropriate and proposes a new categorisation. Section 4.2 explains the data sets created to represent several different drifts. Section 4.3 presents a discussion and a summary of the contributions of this chapter.

4.1 Concept Drift Categorisation

Different types of concept drift can be identified in the concept drift literature. For example, in Gama et al. (2004), several artificial data sets are considered to contain abrupt or gradual drifts. The well known SEA Concepts (Street and Kim; 2001) and STAGGER Concepts (Schlimmer and Fisher; 1986) data sets are considered to have a gradual drift and abrupt drifts, respectively. According to Narasimhamurthy and Kuncheva (2007), gradual changes have also been called gradual drifts, evolutionary changes or simply concept drifts. Abrupt changes have also been called substitutions, concept substitutions, revolutionary changes, abrupt drifts or concept shifts. Moreover, there are also changes called recurring trends or recurring contexts and population drifts.

Besides the existence of different denominations to the same types of drift, the literature does not contain a clear and systematic categorisation of different drifts. Very few and highly heterogeneous categories are considered, separating drifts according to only two criteria (speed

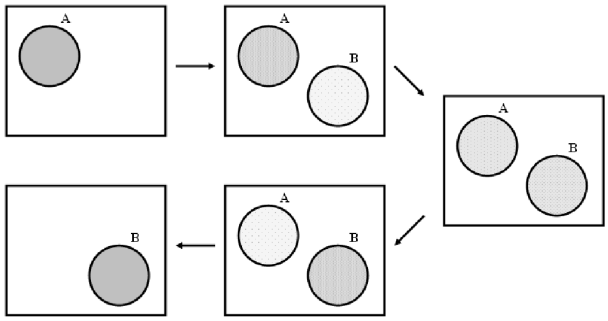
and recurrence). For example, consider a binary classification problem in which the training examples are of the form $[x, y, t]$, where x and y are input attributes representing a point in a unit square and t is a target class indicating whether the point (x, y) is inside or outside a circle represented by the equation $(x-a)^2 + (y-b)^2 \leq r$. Then, consider the concepts represented by the following situations:

- Concept A, in which $a = 0.3$, $b = 0.3$, $r = 0.2$.
- Concept *I1*, in which $a = 0.4$, $b = 0.4$, $r = 0.2$.
- Concept *I2*, in which $a = 0.5$, $b = 0.5$, $r = 0.2$.
- Concept *I3*, in which $a = 0.6$, $b = 0.6$, $r = 0.2$.
- Concept B, in which $a = 0.7$, $b = 0.7$, $r = 0.2$.

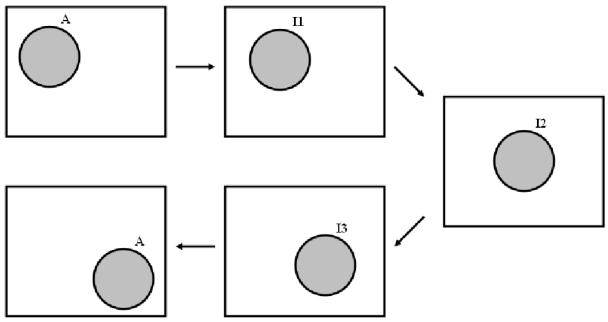
We could have the following drifts:

- New concept B gradually takes over concept A (represented by the schematic figure 4.1(a)):
  1. From time step $t = 1$ to N, concept A is active and the others are inactive.
  2. From time step $t = N + 1$ to $N + drifting\_time$, there is a period of instability. During this period, an example presented to the system belongs to concept A or B with probability $v_a(t)$ or $v_b(t)$, respectively, where $v_a(t) = 1 - v_b(t)$ and $v_b(t) = (t - N)/drifting\_time$.
  3. From time step $t = N + drifting\_time + 1$ to $2N$, concept B is active and the others are inactive.
- Old concept A moves in the direction of concept B by becoming concepts *I1*, *I2* and *I3* (represented by the schematic figure 4.1(b)):
  1. From time step $t = 1$ to $2N/5$, concept A is active and the others are inactive.
  2. From time step $t = 2N/5+1$ to $4N/5$, concept *I1* is active and the others are inactive.
  3. From time step $t = 4N/5+1$ to $6N/5$, concept *I2* is active and the others are inactive.
  4. From time step $t = 6N/5+1$ to $8N/5$, concept *I3* is active and the others are inactive.
  5. From time step $t = 8N/5 + 1$ to $2N$, concept B is active and the others are inactive.

In the literature, these two cases can be considered to represent one gradual drift. The drift in figure 4.1(a) can be considered gradual because the new concept starts gradually to take over, while the drift in figure 4.1(b) can be considered gradual because the old concept gradually moves in the direction of the new concept, creating intermediate concepts. However, these drifts are very different from each other.



(a) New concept gradually takes over.



(b) Old concept gradually moves in the direction of new concept.

Figure 4.1: Heterogeneous categories – gradual drifts. Grey color represents target class 1 and white color represents target class 0. The unconditional pdf remains fixed.

Moreover, if the concepts I1-3 in figure 4.1(b) are considered intermediate concepts, the drift is considered gradual. However, as there is no definition of intermediate concept in the literature, the category "gradual drift" is vague and we could consider that there are four abrupt drifts, instead of one gradual drift.

A clear and systematic categorisation is urgently needed in order to understand different drifts and compare algorithms in a unified framework. The categorisation should divide drifts into different types, according to different criteria, creating mutually exclusive and non-vague categories considering a particular criterion. Each criterion should allow consistent characterization of drifts according to a specific feature, instead of representing different features. Such a categorisation is proposed in this chapter, seeking inspiration from the dynamic optimisation problems area (Branke; 1999, 2002; Yaochu and Branke; 2005). Measures to characterize drifts according to the criteria are also suggested.

This section is further organized as follows. Section 4.1.1 explains the criteria used by the categorisation and suggests measures to characterize drifts. Section 4.1.2 gives additional explanation about how to use the proposed categorisation, including comments on the use of the term intermediate concept.

4.1.1 Criteria and Categories

When analysing a drift in isolation, we can observe that different drifts can cause different amounts of change and take more or less time to be completed. For example, in an information filtering system, a reader may slowly change her/his subjects of interest from a particular subject to another or may quickly get interested in an additional subject. The new subject could be very different or could have some similarities to previous subjects, causing more or less changes to the concept. So, in order to categorise a particular drift, the criteria severity (amount of changes that a new concept causes) and speed (the inverse of the time taken for a new concept to completely replace the old one) are used in the proposed categorisation, as shown in table 4.1.

Although there are applications in which drifts may be completely random and without any pattern, there are also applications in which drifts have certain tendencies. Using again the information filtering example, a reader may not change her/his general preferences all the time without any tendency, so drifts may be less frequent. Besides, a reader could start reading new different subjects and afterwards loose interest for these new subjects, returning to previous concepts (Forman; 2006). In weather forecast, electricity consumption prediction or market basket analysis, there may be drifts which happen depending on the time of the year, having periodic, recurrent and predictable behaviours. The same may happen for prediction of bacterium resistance to antibiotics (Tsymbal et al.; 2006), in which some interesting findings were done about bacterium resistance, such as seasonal context recurring with winter and spring models. So, in order to categorise a sequence of drifts, the proposed categorisation uses the criteria predictability (whether the drifts are completely random or follow a pattern), frequency (how frequently drifts occur and whether they have a periodic behaviour) and recurrence (possibility to return to old concepts), as shown in table 4.1.

This section is further organized as follows: section 4.1.1.1 explains the criteria to categorise drifts in isolation and section 4.1.1.2 explains the criteria to categorise drift sequences. The proposed categories are summarized in table 4.1.

4.1.1.1 Criteria to Categorise Drifts in Isolation

The criterion severity has not been used in the literature yet. It is further divided into class and feature severity. According to class severity, drifts can be divided into severe and intersected. Drifts are severe when no example maintains its target class in the new concept, i.e., 100% of the input space changes the target class when the drift occurs. If part of the input space has the same target class in the old and new concepts, the drift is intersected. For example, the following two measures can be used to characterize drifts according to class severity:

1. Percentage of the input space which has its target class changed after the drift is complete.

Table 4.1: Concept Drift Categorisation.

<table>
<tr><th colspan="3">Criteria</th><th colspan="2">Categories</th></tr>
<tr><td rowspan="5">Drift in Isolation</td><td rowspan="3">Severity</td><td rowspan="2">Class</td><td colspan="2">Severe</td></tr>
<tr><td colspan="2">Intersected</td></tr>
<tr><td rowspan="2">Feature</td><td colspan="2">Severe</td></tr>
<tr><td colspan="2">Intersected</td></tr>
<tr><td colspan="2" rowspan="3">Speed</td><td colspan="2">Abrupt</td></tr>
<tr><td rowspan="2">Gradual</td><td>Probabilistic</td></tr>
<tr><td>Continuous</td></tr>
<tr><td rowspan="5">Drift Sequences</td><td colspan="2" rowspan="2">Predictability</td><td colspan="2">Predictable</td></tr>
<tr><td colspan="2">Non Predictable</td></tr>
<tr><td colspan="2" rowspan="2">Frequency</td><td colspan="2">Periodic</td></tr>
<tr><td colspan="2">Non-Periodic</td></tr>
<tr><td colspan="2" rowspan="3">Recurrence</td><td rowspan="2">Recurrent</td><td>Cyclic</td></tr>
<tr><td>Unordered</td></tr>
<tr><td colspan="2">Not Recurrent</td></tr>
</table>

For example, in the drift represented by figure 4.2, $2/4 = 50\%$ of the input space has its target class changed.

2. Maximum percentage of the input space associated to a particular class in the old concept that has its target class changed in the new concept. For example, in figure 4.2, if we consider the grey area as the input space associated to class 1 and the white area as the input space associated to class 0, class 1 has 100% of its original input space changed, while class 0 has $1/3 \approx 33\%$ of its associated input space changed. So, the maximum percentage is 100%.

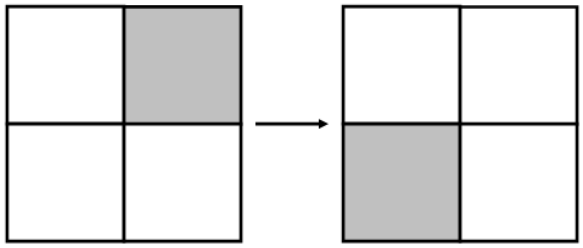


Figure 4.2: Example of an intersected drift. Grey color represents target class 1 and white color represents target class 0.

Although class severity can reflect changes in the prior and posterior probabilities, it does not reflect well changes in the unconditional and class-conditional pdfs. In order to cover these changes, the feature severity criterion, which represents the amount of changes in the distribution of the input attributes, is introduced in the proposed categorisation. In the same way as with class severity, drifts can be severe if the probabilities associated to the whole input space are modified and intersected if part of the input space maintains the same probability. This

criterion can be measured, for example, by calculating the area between the curves of the old and new unconditional pdfs. Another possible measure, which may be easier to calculate, but less descriptive, would be the percentage of the input space which has its probability modified.

The criterion *speed* has been previously used in the literature, although dividing drifts into vague and heterogeneous categories. Speed is the inverse of the drifting time, which can be measured as the number of time steps taken for a new concept to completely replace the old one[1]. So, a higher speed is related to a lower number of time steps and a lower speed is related to a higher number of time steps. According to speed, drifts can be categorised as either abrupt, when the complete change occurs in only one time step, or gradual, otherwise.

As explained by Widmer and Kubat (1996), sometimes concepts will change gradually, creating a period of uncertainty between stable states. The new concept only gradually takes over and some examples may still be classified according to the old concept. An example given by the author is the behaviour of a device beginning to malfunction – it first fails (classifies in a new way) only sometimes, until the new failure mode becomes dominant. We will refer to this type of drift as probabilistic gradual drift.

Another possibility, not considered by Widmer and Kubat (1996), but used in a data set created by Nishida and Yamauchi (2007a), is that the concept itself gradually and continuously changes from the old to the new concept, by suffering modifications between every consecutive time step. We will refer to this type of drift as continuous gradual drift. We use the word *continuous* here to refer to changes in which the old concept suffers modifications at **every** time step since the drift started until the new concept is obtained.

As suggested by Widmer and Kubat (1996), the speed of a drift can be modelled by a function representing the degree of dominance of a concept over the other. This idea was further adopted by other authors (Baena-García et al.; 2006; Narasimhamurthy and Kuncheva; 2007). Figure 4.3 shows an example of a drift which takes 100 time steps. In this example, the concept changes linearly and gradually from the old to the new one.

We will consider here that the speed of a drift can also be modelled by a function representing the changes in the old concept, allowing the representation of continuous gradual drifts. For example, consider a moving hyperplane problem:

$$\sum_i a_i x_i \leq a_0 \ .$$

A continuous change from the old to the new concept could be represented by the function:

[1] A complete replacement of the old concept means that the examples are generated according to the new concept, and yet the old and new concepts can be intersected.

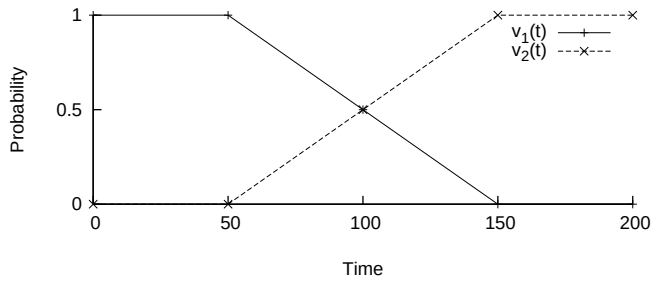


Figure 4.3: Example of a gradual drift with drifting time of 100 time steps. The functions $v_1(t)$ and $v_2(t)$ represent the probability that an example from the old and new concept, respectively, will be presented.

$$a_0(t) = a_0(t-1) * 0.001,\ t_d \leq t \leq t_d + drifting\_time\ ,$$

where t_d is the time step in which the drift starts to happen and $drifting\_time$ is the number of time steps for the new concept to completely replace the old concept.

4.1.1.2 Criteria to Categorise Drift Sequences

Even though a criterion corresponding to predictability has been used in the dynamic optimisation problems area (Branke; 2002; Branke et al.; 2005), predictability has not been used in the concept drift literature. According to this criterion, drifts can be divided into random and predictable. For example, Araujo and Merelo (2007) observed that it is possible to predict changes in the number of times that a certain topic appears in a stream of documents. Even though the study was done in the dynamic optimisation problems area, this could be clearly also considered as a problem in the concept drift area.

Recurrence is a criterion that has been previously used in the literature (Narasimhamurthy and Kuncheva; 2007). It divides drifts into recurrent, if there are returns to previous concepts, and non-recurrent, otherwise. Recurrent drifts can have cyclic or unordered (non cyclic) behaviour. An example of environment with cyclic concepts would be weather forecast, in which different seasons may cause drifts in a cyclic way. An example of problem with unordered recurrent concepts would be market basket analysis, in which a concept drift can be caused by the introduction of a new product. However, after a certain amount of time, the customers can find that the new product is not so good and the concept can return to the previous one. It is worth to note, though, that there may be a return to a concept which is similar to a previous one, but not the same. So, a return to a previous concept may be associated to a certain severity in relation to the previously existing concept.

Frequency is another criterion that has not been used in the literature yet. It can be

measured, for example, by the number of time steps between the start of two consecutive drifts. According to this criterion, drifts can be divided into periodic and non-periodic. If a new concept drift starts to take place at every t time steps, the drifts are considered periodic. Otherwise, they are considered non-periodic.

It is worth to note that, although recurrence and frequency are distinct criteria, several drifts that present a recurrent behaviour are also periodic. For example, in weather forecast, drifts can happen periodically according to the time of the year, returning to previous concepts depending on the season.

4.1.2 Intermediate Concepts Term Revocation

Besides defining the criteria and categories, in order to provide a proper categorisation, it is necessary to consider if a concept could be denominated as an intermediate concept between the old and the new concept or not. As briefly explained before, consider the concepts represented in figure 4.1(b). Some authors would regard the concepts represented by the circles between A and B as intermediate concepts between the old concept A and the new concept B.

Depending on the definition of intermediate concept, both the category to which a particular drift belongs and the characterization of the drift according to a particular criterion can change. For example, if we consider I1-3 as intermediate concepts in figure 4.1(b), there is only one drift (from A to B), which has 100% of severity according to the second measure of severity (M.2). However, if we do not consider I1-3 as intermediate concepts, we would have 4 intersected (not so severe) drifts. If we consider I1-I3 as intermediate concepts active for 100 time steps each and the complete change from a concept to another takes only 1 time step, the drift would be considered gradual. However, if I1-3 are not considered intermediate concepts, we would have 4 abrupt drifts.

A possible definition for intermediate concept could be a concept which is not active for enough time to be PAC (Valiant; 1984) learnt. In Kuh et al. (1990), an upper bound for the maximum number of time steps required for a new concept to be PAC learnt, considering that the previous concept was PAC learnt, is defined. However, such upper bound depends on the VC-dimension (Vapnik and Chervonenkis; 1971) of model and a definition of intermediate concept based on this bound would depend on the class of model, not having straightforward application.

Another option for defining intermediate concepts would be to consider it as a subjective idea. In this way, before using a particular data set which contains drifts, it would be necessary to subjectively consider a concept as intermediate or not. However, a subjective definition would affect the categorisation of drifts and the same category could have very different characteristics depending on the characteristics of the intermediate concepts.

So, in order to provide a simpler and more straightforward categorisation, the use of the term intermediate to refer to concepts is not permitted when using the proposed categorisation. Every concept should be considered as either old or new and the notion of old and new concepts should only be applied for a particular drift in isolation. We can eliminate the use of the term intermediate concept thanks to the inclusion of criteria not previously used in the literature, such as predictability and severity. These criteria allow drifts to be well described and differentiated even without using this term, as described in section 4.1.1, making the term unnecessary. For example, we can name the drifts in figure 4.1(b) as a sequence of four intersected drifts, instead of one drift with three intermediate concepts. There is no need to name the concepts I1-3 as intermediate if we know that the drifts are in a sequence of intersected drifts.

It is important to note that, in order to describe drifts, all the proposed criteria should be considered at the same time. For example, a particular drift can be intersected, abrupt and in a sequence of periodic, random and non-recurrent drifts. Real world problems may present more than one drift and each drift can be of a different type. In this way, we create a more systematic, well-defined, non-heterogeneous and mutually exclusive categorisation than that currently used in the literature, which considers only the criteria speed and recurrence.

4.2 Artificial Data Sets

This section presents a data sets generator that can be used to create several different types of drift according to the categorisation (section 4.2.1) and the artificial data sets used in the thesis (section 4.2.2).

When working with real world data sets, it is often not possible to know exactly when a drift starts to occur, which type of drift is present or even if there really is a drift. So, it is not possible to perform a detailed analysis of the behaviour of algorithms in the presence of concept drift using only pure real world data sets. In order to analyse the strong and weak points of a particular algorithm, it is necessary first to check its behaviour using artificial data sets containing simulated drifts.

Depending on the type of drift in which the algorithm is weak, it may be necessary to adopt a different strategy to improve it, so that its performance is better when applied to real world problems. In the same way, when analysing strategies that may be adopted in an algorithm to handle concept drift, it is necessary to know in which situations and how these strategies could contribute, i.e., it is necessary to know with which sort of drifts they could help. After checking the behaviour of concept drifting handling approaches with artificial data sets, it is still important to use real world data sets in order to further confirm the usefulness of the approaches.

4.2.1 Generator

Artificial data sets such as STAGGER Boolean concepts (Schlimmer and Granger Jr.; 1986) and SEA moving hyperplane concepts (Street and Kim; 2001) do not present enough variety of drifts to perform principled studies. In order to allow analyses considering several types of drift with different amounts of severity and speed, a data sets generator was developed in the present work. The generator is inspired on existing benchmarks such as STAGGER and SEA, but allows the generation of different types of drift for four problems, according to the equations presented in table 4.2. In the equations, for all the problems but Boolean, a, b, c, d, r and a_i can assume different numeric values and eq can represent an operator such as $\leq$ or $>$ in order to define different concepts. The data examples contain x/x_i and y as the input attributes and the concept (which can assume value 0 or 1) as the output attribute.

Table 4.2: Artificial Data Sets.

Problem	Equation	Fixed Values	Before→After Drift	Severity
Circle	$(x-a)^2+(y-b)^2 eq\ r^2$	$a=0.5$	$r=0.2 \rightarrow 0.3$	16%
		$b=0.5$	$r=0.2 \rightarrow 0.4$	38%
			$r=0.2 \rightarrow 0.5$	66%
SineV	$y\ eq\ a\,sin(bx+c)+d$	$a=1$	$d=-2 \rightarrow 1$	15%
		$b=1$	$d=-5 \rightarrow 4$	45%
		$c=0$	$d=-8 \rightarrow 7$	75%
SineH		$a=5$	$c=0 \rightarrow -\pi/4$	36%
		$d=5$	$c=0 \rightarrow -\pi/2$	57%
		$b=1$	$c=0 \rightarrow -\pi$	80%
Line	$y\ eq\ -a_0+\sum_{i=1}^{d} a_i x_i$	$d=1$	$a_0=-0.4 \rightarrow -0.55$	15%
		$a_1=0.1$	$a_0=-0.25 \rightarrow -0.7$	45%
			$a_0=-0.1 \rightarrow -0.8$	70%
Plane		$d=2$	$a_0=-2 \rightarrow -2.7$	14%
		$a_1=0.1$	$a_0=-1 \rightarrow -3.2$	44%
		$a_2=0.1$	$a_0=-0.7 \rightarrow -4.4$	74%
Bool	$(color\ eq_1\ a\ op_1\ shape\ eq_2\ b)\ op_2\ size\ eq_3\ c$	$c=S \vee M \vee L$	$a=R, op_1 \wedge b=R \rightarrow R \vee T$	11%
		$op_2 \wedge$	$a=R, b=R, op_1 \wedge \rightarrow \vee$	44%
		$eq_{1,2,3}=$	$a=R \rightarrow R \vee G, b=R \rightarrow R \vee T, op_1 \wedge \rightarrow \vee$	67%

The Boolean problem is inspired by the STAGGER problem (Schlimmer and Granger; 1986), but it allows the generation of different drifts, with different levels of severity and speed. In this problem, each example has three input attributes: color (red R, green G or blue B), shape (triangle T, rectangle R or circle C) and size (small S, medium M or large L). The concept is then represented by the Boolean equation given in table 4.2, which indicates the color, shape and size of the objects which belong to class 1 (true). In that expression, a represents a conjunction

or disjunction of different possible colors, b represents shapes, c represents sizes, eq represents $=$ or $\neq$ and op represents the logical connective $\wedge$ or $\vee$. For example, the first concept of the Boolean data set which presents a drift with 11% of severity in table 4.2 is represented by:

$$(color = R \ \wedge shape = R) \wedge size = S \vee M \vee L \ .$$

and the second concept of this drift is represented by:

$$(color = R \ \wedge shape = R \vee T) \wedge size = S \vee M \vee L \ .$$

4.2.2 Data Sets Used in the Thesis

The experiments done in the thesis use several data sets for each of the following six problems, created according to the generator: circle, sine moving vertically, sine moving horizontally, line (moving hyperplane with $d = 1$), plane (moving hyperplane with $d = 2$) and Boolean. Eight irrelevant attributes and 10% class noise were introduced in the plane data sets.

Each data set contains one drift and different drifts were simulated by varying among three amounts of severity (as shown in table 4.2) and three speeds, thus generating nine different drifts for each problem. The feature severity affects the same areas of the input space as the class severity for all the problems but plane and Boolean. So, we will refer to class severity simply as severity in the experiments. For plane and Boolean, there is no feature change. The speed was modelled by the following linear degree of dominance functions:

$$v_n(t) = \frac{t - N}{drifting\_time}, \ N < t \leq N + drifting\_time$$

and

$$v_o(t) = 1 - v_n(t), \ N < t \leq N + drifting\_time \ ,$$

where $v_n(t)$ and $v_o(t)$ are the degrees of dominance of the new and old concepts, respectively; t is the current time step; N is the number of time steps before the drift started to occur; and $drifting\_time$ varied among 1, $0.25N$ and $0.50N$ time steps.

The data sets are composed of $2N$ examples. The first N examples belong to the old concept ($v_o(t) = 1$, $1 \leq t \leq N$), where $N = 1000$ for circle, sineV, sineH and line and $N = 500$ for plane and Boolean. The next $drifting\_time$ examples ($N < t \leq N + drifting\_time$) were generated according to the degree of dominance functions, $v_n(t)$ and $v_o(t)$. The remaining examples belong to the new concept ($v_n(t) = 1$, $N + drifting\_time < t \leq 2N$).

The range of x or x_i was $[0, 1]$ for circle, line and plane; $[0, 10]$ for sineV; and $[0, 4\pi]$ for sineH. The range of y was $[0, 1]$ for circle and line, $[-10, 10]$ for sineV, $[0, 10]$ for sineH and $[0, 5]$ for plane. For plane and Boolean, the input attributes are normally distributed through the whole input space. For the other problems, the number of instances belonging to class one and zero is always the same, having the effect of changing the unconditional pdf when the drift occurs.

4.3 Summary and Discussion

The contributions of this chapter are a new concept drift categorisation and new artificial data sets representing several different types of drift.

The literature lacks a clear and systematic categorisation of different drifts. Currently, very few and highly heterogeneous categories are used, separating drifts according to only two criteria (speed and recurrence). The criterion speed used in the literature represents at the same time more than one feature of drifts, so that very different drifts can be considered of the same type. Besides, the literature does not consider the existence of certain features of drifts, such as frequency and predictability.

The main contribution of this chapter is a new concept drift categorisation. The categorisation divides drifts into different types, according to several different criteria: severity, speed, predictability, frequency and recurrence. Each criterion allows consistent characterization of drifts according to a specific feature, instead of representing different features at the same time. The use of the term "intermediate" to refer to concepts is not allowed. In this way, mutually exclusive and non-vague categories are created.

The categorisation and in particular its quantifications are mainly targeted to artificial data sets, as we usually cannot know exactly what types of drift are present in real world data sets. As explained in section 4.2, it is important to use not only real world, but also artificial data sets in studies of drift, so that we can know the types of drift to which the approaches behave better/worse. The use of artificial data sets not only provides a better understanding of the approaches' behaviour and when they are likely to perform well, but also aids the proposal of solutions to improve their performance.

Chapters 5 and 6 further illustrate the usefulness of the categorisation experimentally. For instance, in section 5.3, we can observe that the impact of severity on the test error is very different from the impact of speed. So, different strategies can be developed in order to handle drifts with different amounts of severity and speed. An approach which is accurate to different amounts of severity may have a very different behaviour considering different amounts of speed. In the literature, the criteria severity and speed are mixed, not allowing proper evaluation of approaches for this case.

Another example of how chapter 6 illustrates the usefulness of the categorisation is when the KDD network intrusion detection data set (Hettich and Bay; 1999) is used. In this case, the high frequency of intercalated recurrent drifts allows approaches which are not prepared for recurrent drifts to achieve a good behaviour. A lower frequency would require additional strategies to deal with the recurrence. So, it is important to use not only the criterion recurrence, but also frequency, which is considered only in the categorisation proposed in this chapter.

The literature does not consider the criterion predictability either. This criterion has shown to be useful in the dynamic optimisation problems area (Branke; 2002; Branke et al.; 2005) and should also be considered in the concept drifts area. An example of use for it is in the prediction of changes in the number of times that a particular topic appears in a stream of documents (Araujo and Merelo; 2007). If a change can be predicted, appropriate measures can be designed for a certain approach to recover quicker and/or be less affected by the change. As predictable and non-predictable drifts are not distinguished in the literature, it would not be possible to correctly evaluate approaches for this case. The proposed categorisation, on the other hand, would allow a proper evaluation.

The benchmarks used in the literature, such as SEA (Street and Kim; 2001) and STAGGER (Schlimmer and Granger Jr.; 1986) concepts, do not contain enough variety of drifts to allow principled and detailed studies. Based on existing benchmarks, a data sets generator is presented in section 4.2.1. It can be used to generate all the types of drift from the proposed categorisation. New data sets simulating drifts with low, medium and high severity and speed were created and are presented in section 4.2.2. They allow more detailed and principled analysis of approaches/strategies in the presence of concept drift and are used in chapters 5 and 6. Sequences of drift are studied directly through the use of real world or semi-real world data sets in the thesis. The thesis does not focus on studies of recurrent and predictable drifts, which are proposed as future work.

Chapter 5

A Diversity Study in the Presence of Drift

Ensembles of learning machines have been widely studied and successfully used in offline mode. As examples, we can cite approaches such as negative correlation learning (Liu and Yao; 1999a,b; Liu et al.; 1999; Chen and Yao; 2009), bagging (Breiman; 1996a) and boosting (Schapire; 1990; Drucker et al.; 1992; Freund; 1995; Freund and Schapire; 1996b,a). The success of ensembles in offline mode inspired their use for online learning (Blum; 1997; Lim and Harrison; 2003; Kotsiantis and Pintelas; 2004; Oza and Russell; 2001a,b, 2005; Fern and Givan; 2000, 2003; Lee and Clyde; 2004) and concept drift (Stanley; 2003; Kolter and Maloof; 2003, 2007; Wang et al.; 2003; Scholz and Klinkenberg; 2005, 2007b; Ramamurthy and Bhatnagar; 2007; He and Chen; 2008).

Although ensembles have been used to handle concept drift, the literature does not contain any deep study of why they can be helpful for that and which of their features can contribute or not to deal with concept drift. A better understanding of the behaviour of ensembles in online changing environments can reveal if their potential is being correctly used and would allow better exploitation of their features.

One of the features which may help dealing with drifts is diversity. As explained in section 1.2.2, in offline mode, diversity among base learners is an issue that has been receiving lots of attention in the ensemble learning literature. Many authors believe that the success of ensemble algorithms depends on both the accuracy and the diversity among the base learners (Dietterich; 1997; Kuncheva and Whitaker; 2003). However, no study of diversity has ever been done in online changing environments.

Section 5.3 presents a diversity study of ensembles in the presence of different types of concept drift, providing a deeper understanding of when, why and how online ensemble learning can help to deal with drifts. This is an answer to the research question presented in section

1.2.2. Section 5.4 extends the diversity study and shows how to use information from the old concept in order to aid the learning of the new concept, providing an answer to the research question presented in section 1.2.3. The underlying online ensemble learning algorithm used in the study is explained in section 5.2 and the statistical method used in the analyses is explained in section 5.1.

5.1 Analysis of Variance (ANOVA)

The analyses presented in section 5.2 and 5.3 use Analysis of Variance (ANOVA) (Montgomery; 2004). ANOVA is a set of statistical methods that can be used to test hypotheses about the effect of different factors on a response variable.

For understanding the general idea of ANOVA, let's consider the simple case in which we have only one factor. For example, consider that we are interested in analysing the effect of the parameter A on a certain algorithm's accuracy using a certain data set. The parameter A is a factor and the accuracy is the response. Consider also that we choose a different values (factor levels) for A and run the algorithm r times for each factor level. So, the total number of response observations used by ANOVA is $N = a * r$.

ANOVA methods are based on partitioning the total variability of the response into several components, which are attributed to different sources of variation. The total variability (SS_T) is measured by the sum of squares over all the observations. In the example considered here, it is decomposed into the variability due to the factor choices ($SS_{Treatments}$) and the variability due to the error (SS_E):

$$SS_T = SS_{Treatments} + SS_E \ .$$

There are different ways to estimate the variabilities. The most frequently used is the Type III (Marginal) Sum of Squares, which represents the additional variability explained by adding the factor of interest.

The statistical analysis can use the null hypothesis that there is no difference in treatment means (no difference in response when using different factor levels) and the alternative hypothesis that there is difference. For testing whether this null hypothesis is true, the F statistic of the test is calculated as:

$$F = \frac{SS_{Treatments}/(a-1)}{SS_E/(N-a)} \ .$$

So, if the variability caused by the factor choices is large in comparison to the variability due

to error, F will be larger and the null hypothesis that there is no difference in treatment means will be rejected. On the other hand, if the variability due to error is large in comparison to the variability due to the factor choice, the null hypothesis will be accepted.

When more than one factor is used, SS_T can be decomposed into the variability due to each factor, due to interaction of factors and due to error. In this way, the effect of each factor and interaction on the response can be analysed.

Factors can be categorized as within-subject or between-subject (Lane et al.; 2008). Within-subject factors involve comparisons of the same subjects under different conditions (factor levels). Between-subject factors are factors in which a different group of subjects is used for each factor level. As the term subject may be difficult to understand in the computer science domain, the examples given by Lane et al. (2008) will be used to illustrate within and between-subject factors.

Consider a study of the treatment of a certain disease using drugs. Each contaminated person (subject)'s performance was measured four times, once after being on each of four drug doses for a week. Therefore, each subject's performance was measured at each of the four levels of the factor "dose", which is a within-subject factor.

Now, consider an experiment conducted for comparing four methods of teaching vocabulary. If a different group of students (subjects) is used for each of the four teaching methods, then teaching method is a between-subjects variable.

When more than one factor is used, it could happen that we have a split-plot (mixed) design, which involves both between-subject and within-subject factors. In this case, ANOVA has to be done in two parts: one for analysing the within-subject effects and the other one for analysing the between-subject effects. As explained in sections 5.2 and 5.3, this is the type of ANOVA used in the thesis.

The main assumption done by split-plot ANOVA is the sphericity (Demšar; 2006). Consider the covariance matrix of the levels of a within-subjects factor. A sufficient (but not necessary) condition for sphericity is that all the covariances are equal and all the variances are equal in the populations being sampled. This sufficient condition is frequently used to give an intuition of what sphericity is. However, the sphericity assumption is a bit less strict (Baguley; 2004). Consider the differences between the responses for each pair of factor level. For example, for a factor A with three different levels, let $A_1(r)$, $A_2(r)$ and $A_3(r)$ be the responses obtained for each treatment (factor level) A_1, A_2 and A_3 on the subject r. For each subject r, calculate the differences $A_1(r) - A_2(r)$, $A_1(r) - A_3(r)$, $A_2(r) - A_3(r)$. Then, calculate the variance for each pair of factor level. The sphericity assumption considers that all the variances of the differences are equal.

If the sphericity assumption is violated, the split-plot ANOVA can get high type I error (reject the null hypothesis when it was true) (Demšar; 2006). Mauchly's tests (Mauchly; 1940) can

be used to detect violations of the sphericity assumption. If violations are detected, corrections such as Greenhouse-Geisser (Greenhouse and Geisser; 1954) can be applied to the ANOVA's p-value so that the type I error will not be increased.

In addition to ANOVA, measures of effect size such as eta-squared (Howell; 2007; Pierce et al.; 2004) can be used to determine whether the effect of a certain factor or interaction is higher than the effect of another factor or interaction. The eta-squared of a certain factor or interaction F is calculated as SS_F/SS_T, where SS_F is the variability due to F. The higher the eta-squared, the greater the relevance of the corresponding factor or interaction on the response. For a split-plot ANOVA, eta-squareds need to be calculated separately within the context of the within-subject effects and the between-subject effects. In this situation, they will sum to 1 for the within-subjects and will sum to 1 for the between-subject effects.

5.2 Modified Online Bagging

This section presents a modified version of Oza and Russell (2001a,b, 2005)'s online bagging and the reasons for choosing it to perform the diversity study presented in this chapter.

In order to study the impact of diversity in changing environments in a principled way, we need an ensemble learning algorithm which allows us to "tune" diversity. As explained in section 2.4, it is also desirable to use a parallel-generation multiple-update approach, as then (1) there is no need to determine when to stop the learning of a particular ensemble member and (2) a single training example can contribute to the learning of several ensemble members, a property which is believed to be essential for obtaining fast convergence to the desired concept (Fern and Givan; 2003).

NCL (Liu and Yao; 1999b,a) is an ensemble learning approach which allows encouraging more or less diversity through the strength of the penalty term, as explained in section 2.2. This approach is also parallel-generation and multiple-update when applied to online learning. However, section 3.2 shows that NCL usually needs to be combined with other approaches, such as online bagging, to obtain higher accuracy in online mode. So, the NCL penalty term is not the only factor which encourages diversity and varying it may not consistently vary diversity.

As explained in section 2.4.1, online bagging is also a parallel-generation multiple-update approach. However, the amount of diversity encouraged during the learning is not controllable. In the present section, a simple modification inspired by Oza and Russell (2001a,b, 2005)'s online boosting is introduced in order to allow "tuning" diversity in online bagging. The modification consists of adopting a user-chosen parameter λ for the distribution $Poisson(\lambda)$ of Oza and Russell (2001a,b, 2005)'s online bagging algorithm (algorithm 2.1), instead of fixing it as $Poisson(1)$.

The reason for choosing online bagging, instead of other algorithms such as online boosting, is that this simple modification allows us to consistently encourage more or less diversity, as λ

is the only source of diversity apart from the data itself. In the remainder of this section, it is shown that higher/lower λ values are associated to lower/higher average diversity. Section 5.2.1 explains the experimental design used to show that and section 5.2.2 explains the analysis results.

5.2.1 Experimental Design

Analysis of variance (ANOVA) (Montgomery; 2004) was used to verify the impact of λ on diversity when using modified online bagging under several different drift conditions. The study used the artificial data sets presented in section 4.2.2. A split-plot (mixed) design, where a between-subjects design is combined with a repeated measures design was adopted.

The between-subject factors are severity and speed, which varied among three different levels each to create different data sets, as explained in section 4.2.2. The within-subject factors are λ and the time step analysed. The factor λ varied among eight different levels in order to encourage several different amounts of diversity: 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1. The factor time varied among four different levels in order to allow analysis right before the drift, shortly after the drift, a bit longer after the drift and longer after the drift: $0.99N$, $1.1N$, $1.5N$ and $2N$, where N is the number of time steps until the beginning of the drift, as described in section 4.2.2. The response (diversity) was measured through the Q statistic (equation 1.1), which is a popular diversity measure for classifiers, at the time step analysed.

Mauchly's tests of sphericity (Mauchly; 1940) detected violations of the sphericity assumption (null hypothesis always rejected with p-value less than 0.001), so Greenhouse-Geisser corrections (Greenhouse and Geisser; 1954) were used. The effect size was evaluated according to the eta-squared value (Howell; 2007; Pierce et al.; 2004), as explained in section 5.1.

Thirty repetitions for each combination of factor level (except time) were done, giving a total of 2160 executions for each artificial problem. The ensembles were composed of 25 Incremental Tree Inducers (ITIs) (Utgoff et al.; 1997), which are online decision trees. Decision trees were chosen for being one of the most used base learners in the concept drift literature (Gama et al.; 2004; Baena-García et al.; 2006; Oza and Russell; 2005; Kolter and Maloof; 2003; Chu and Zaniolo; 2004; Gao, Fan, Han and Yu; 2007; Wang et al.; 2003; Gao, Fan and Han; 2007; Street and Kim; 2001).

5.2.2 Analysis

The results of the ANOVA test of within-subject effects are shown in table 5.1, together with the eta-squared values. The table presents the type III sum of squares (SS), the degrees of freedom (DF), the mean squares (MS), the F test statistics (F) and the eta-squared values (Eta). Interactions between 2 factors are represented by factor1*factor2. P-values less than

0.01 represent rejection of the null hypothesis that the average response is statistically equal at all the levels of the corresponding factors, considering significance level of 1%. Only the factors/interactions with eta-squared higher than 0.10 are shown, in order to facilitate reading, and their p-values are always less than 0.001. We can observe that λ always has the largest effect size. The effect is very large – eta-squared is always more than 0.90 and usually more than 0.97. The eta-squared values for the other factors and interactions are very low, usually below 0.015.

Considering the between-subject tests (table 5.2), severity usually has considerably large effect size (between 0.10 and 0.60). The eta-squared values associated to speed and the interaction severity*speed are usually lower than 0.01.

Table 5.1: ANOVA – Test of Within-Subjects Effects on the Q Statistic: Factors/interactions with eta-squared higher than 0.10, in decreasing order of effect size for each problem. The p-value for these is always less than 0.001.

Problem	Factor/Interaction	SS	DF	MS	F	Eta
Circle	λ	1150.067	2.837	405.325	51414.895	0.974
SineV	λ	1460.340	2.734	534.204	63066.663	0.974
	λ * Time	17.360	4.024	4.314	3975.454	0.012
SineH	λ	1040.883	3.999	260.288	92479.771	0.976
	Time	12.555	1.241	10.114	54507.625	0.012
Line	λ	1466.008	2.556	573.532	52628.413	0.973
	λ * Time	17.557	3.593	4.886	3750.511	0.012
Plane	λ	742.586	3.661	202.813	13339.739	0.959
	λ * Sev	10.866	7.323	1.484	97.598	0.014
Bool	λ	1112.990	2.429	458.207	7974.545	0.911
	λ * Time	34.453	5.067	6.800	3059.544	0.028
	Time	25.778	1.330	19.388	15596.960	0.021

Table 5.2: ANOVA – Test of Between-Subjects Effects on the Q Statistic: Factors/interactions with eta-squared higher than 0.10, in decreasing order of effect size for each problem. All of these had p-value less than 0.001.

Problem	Factor/Interaction	SS	DF	MS	F	Eta
Circle	Sev	0.518	2	0.259	71.751	0.352
SineV	Sev	0.153	2	0.076	17.844	0.118
SineH	Sev	0.169	2	0.084	45.851	0.250
Plane	Sev	2.992	2	1.496	183.376	0.560
Bool	Sev	1.055	2	0.527	17.461	0.117

So, the factors with more influence on the response are λ and severity. That means that the parameter λ of online bagging has indeed very large influence on the ensemble's diversity. The factor severity, which is related to the data being presented, also has some influence.

As we will need to analyse the interactions λ*time*severity in the next sections, all the plots of marginal means for λ vs severity and λ vs speed for each time step analysed were generated. The plots show the average Q statistic associated to each one of the combinations among these factors. As the number of plots is very large (192 plots) and they all present the same tendencies, they were omitted and only the plot of the main effect of λ on the response for circle is shown (figure 5.1).

The plots of marginal means show that higher λ values generate higher Q statistics (lower diversity) and lower λ values generate lower Q statistics (higher diversity). This behaviour is reflected on the main effect of λ on the response (figure 5.1).

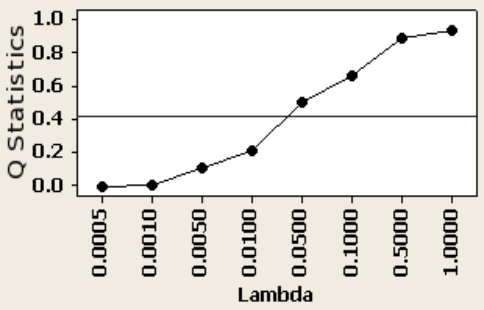


Figure 5.1: Plot of the main effect of λ on Q statistic for circle.

Besides, the lowest Q statistic generated is always close to 0, whereas the highest is always close to 1 (although always lower than 1). Frequently, the Q statistic values generated for $\lambda = 0.0005$ and 0.001 are more similar to each other. The same happens for $\lambda = 1$ and 0.5. Different amounts of severity or speed usually do not change much the response in comparison to λ and never cause a higher λ to have lower Q statistic than a lower λ.

5.2.3 Discussion

The parameter λ of the modified online bagging algorithm behaves in the way we desire in order to allow a principled study of diversity in the presence of concept drift. Among the factors analysed, λ is the one with the most significant effect on diversity, so that varying it allows us to consistently "tune" diversity. Higher/lower λ values generate higher/lower Q statistics (lower/higher diversity).

5.3 The Influence of Diversity on the Old/New Concept

This section presents an analysis of the impact of diversity on the learning of the old or the new concept. No special feature to deal with drifts is used, so that the impact of diversity by itself can be analysed. The following points are checked:

1. The differences between the influence of diversity on the ensemble error on the old and new concepts (section 5.3.2.1).

2. Whether different types of drift require different amounts of diversity to get low error on the new concept (section 5.3.2.1).

3. The influence of diversity on the ensemble's sensitivity to drifts and on the adaptation to the new concept considering base learners that learnt the old concept (section 5.3.2.2).

4. Whether it is possible to exploit diversity to better handle concept drift (section 5.3.2.2).

In this way, this section provides an answer to the research question presented in section 1.2.2: "when, how and why can ensembles be helpful for dealing with drifts?" The study also demonstrates the different effects of severity and speed of drifts, further confirming the importance of the drifts categorisation proposed in section 4.1.

The analysis indicates that, before the drift, ensembles with less diversity obtain lower test errors. On the other hand, it is a good strategy to maintain highly diverse ensembles to obtain lower test errors on the new concept shortly after the beginning of the drift independent of the type of drift, even though high diversity was more important for more severe drifts. Longer after the drift, high diversity becomes less important. The experiments showed that diversity by itself helped to reduce the initial increase in error caused by a drift, but did not provide a fast recovery from drifts in the long term. The effect of severity on the error on the new concept was higher than the effect of speed.

The rest of this section is organized as follows. Section 5.3.1 explains the experimental design and measures used for the analysis. Section 5.3.2 presents the analysis itself. Section 5.3.3 gives a discussion of the results.

5.3.1 Experimental Design

The study uses the artificial data sets explained in section 4.2.2. As we will see in section 5.3.2, drifts can benefit from high diversity during the learning of the old concept independent of the severity and speed. So, we can perform additional experiments using real world data sets with simulated drifts to further confirm the analysis. In this further analysis, it is important for us to know that there is a drift and when the drift starts in order to obtain useful knowledge, but it is not essential to know all the details about the type of drift.

The additional data sets are based on the databases Contraceptive, Yeast, Iris and Car, from the UCI Machine Learning Repository (Newman et al.; 2010) and they will be referred to as UCI data sets in the thesis. Contraceptive and Yeast represent more difficult databases, whereas Iris and Car are easier. Simulated drifts inspired by Scholz and Klinkenberg (2005) were introduced, creating data sets which have both real world and artificial features. They contain

difficult or unknown decision boundaries at the same time as some of the features of the drifts are known.

The order of examples of the car database was randomized and the examples were divided into 3 partitions in order to simulate 2 drifts. The first partition represents the examples before any drift; the second partition represents examples after the beginning of the first drift and before the second drift; and the third partition represents examples after the beginning of the second drift. The other databases were replicated 3 (or 4, for yeast) times, making them 3 (or 4) times bigger. Each replication was used to create a partition and the order of examples of each partition was randomized. Different concepts were created by changing the labels of the target classes of the examples according to table 5.3. As we can observe from the table, all the second drifts present a partial return to the first concept, i.e., part of the changes caused by the first drift are undone by the second drift.

Table 5.3: UCI Data Sets - Total number of time steps used for learning; rounded percentage of examples of each class in the original database; and target classes C_i used to create the drifting data sets in each partition i. The target class given in each cell represents the label given to all the examples whose original class is specified in the corresponding row.

Problem and Time Steps	Original Class	C1	C2	C3	C4
Contraceptive	No-use 42.70%	1	2	2	-
3314	Long-term 22.61%	2	3	1	-
	Short-term 34.69%	3	1	3	-
Yeast	CYT 31.16%	1	2	1	1
4452	NUC 28.87%	2	1	2	2
	MIT 16.42%	3	4	4	4
	ME3 10.97%	4	3	3	3
	ME2 3.43%	5	5	5	6
	ME1 2.96%	6	6	6	5
	EXC 2.49%	7	7	7	8
	VAC 2.02%	8	8	8	7
	POX 1.35%	9	9	9	10
	ERL 3.36%	10	10	10	9
Iris	Setosa 33.33%	1	4	4	-
337	Versicolour 33.33%	2	1	2	-
	Virginica 33.33%	3	3	3	-
	0%	4	2	1	-
Car	Unacc 70.02%	0	1	0	-
1296	Acc 22.22%	1	0	1	-
	Good 3.99%	1	0	0	-
	Very good 3.76%	1	0	0	-

Each partition has size N ($N = 1473$ for contraceptive, 1482 for yeast, 150 for iris and 576 for car). The $drifting\_time$ of the first drift is $0.25N$ for contraceptive and yeast and 1 for iris and car. The $drifting\_time$ of the second (and third) drift is 1 for contraceptive, $0.25N$ for

yeast and $0.5N$ for iris and car. The speed is modelled by linear degree of dominance functions, similarly to the data sets explained in section 4.2.2.

The experiments were planned as follows. ANOVA was used to analyse the impact of λ and other factors on the test error to get answers to the points (1) and (2) (section 5.3.2.1). After that, the test error over time was analysed in order to check how sensitive to drifts ensembles with different diversity levels are and how fast they recover from drifts (section 5.3.2.2), getting answers to points (3) and (4). Finally, the results obtained using the UCI data sets are presented, reassuring the analysis (section 5.3.2.3).

As in section 5.2, the ANOVA analysis with the artificial problems used a split-plot (mixed) design, where a between-subjects design is combined with a repeated measures design. The ANOVA analysis with the UCI problems used a repeated measures design (Montgomery; 2004), as there are no factors severity and speed in this case. The response was the test error.

Several authors use different ways to test their algorithms in online and incremental learning. Some authors create a test set which reflects exactly the underlying distribution of the train data at the time step to be tested (Street and Kim; 2001; Kolter and Maloof; 2003). Other authors use the average prequential error, calculated by updating the average with the prediction of each example before its learning (Gama et al.; 2004; Baena-García et al.; 2006). Others argue that the test error should not reflect exactly the underlying distribution of the train data, as, in the real world, the distribution of the problem may change since the presentation of the last training examples (Gao, Fan and Han; 2007). Besides, in incremental learning, the algorithms are frequently tested with the next chunk of data to be learnt, before its learning (Scholz and Klinkenberg; 2007a; Tsymbal et al.; 2006).

In this section, we are interested on the effect of diversity on the learning of the old/new concepts. So, for the artificial data, test sets were created using two partitions. The first partition is composed of $0.25N$ examples generated according to the old concept and the second is composed of $0.25N$ examples generated according to the new concept. In order to test the system before the drift, only the examples belonging to the first concept are used, whereas to test the system after the beginning of the drift, only the examples of the new concept are used, even if the drift is still not completed. A similar procedure was adopted for the real world data, but the test sets were created by extracting the last 25% examples of each partition that composes the data sets.

The between-subject factors used in the split-plot design are severity and speed, which varied among three different levels each and were used to create the artificial data sets, as explained in section 4.2.2. The within-subject factors used in both designs are the parameter λ from the modified online bagging and the time step analysed. The factor λ varied among eight different levels: 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1 and was used to encourage different amounts of diversity. The factor time step varied among $0.99N$, $1.1N$, $1.5N$ and $2N$ for the artificial data sets and also included $2.1N$, $2.5N$, $3N$ (and $3.1N$, $3.5N$ and $4N$ for yeast) for the

UCI data sets. These time steps were chosen in order to analyse the response shortly before the drift, shortly after the drift and longer after the drift.

Mauchly's tests of sphericity (Mauchly; 1940) detected violations of the sphericity assumption (p-value always less than 0.001), so Greenhouse-Geisser corrections (Greenhouse and Geisser; 1954) were used. As in section 5.2, eta-squared (Howell; 2007; Pierce et al.; 2004) was used to measure the effect size.

Thirty repetitions for each combination of factor level (except time) were done, giving a total of 2160 executions for each artificial problem and 240 executions for each UCI problem. The ensembles were trained with the modified online bagging algorithm presented in section 5.2 and each ensemble was composed of 25 ITI online decision trees (Utgoff et al.; 1997), as in section 5.2.

5.3.2 Analysis

5.3.2.1 Test Error Before and After a Drift

In this section, we concentrate mainly on checking the points (1) and (2) mentioned in the beginning of section 5.3: (1) the differences between the influence of diversity on the ensemble error on the old and new concepts and (2) whether different types of drift require different amounts of diversity. In order to do so, ANOVA was used to analyse the influence of each factor mentioned in section 5.3.1 on the response, which is the test error, as explained in section 5.3.1. The data sets used in this analysis are the artificial data sets presented in section 4.2.2. We will consider the influence of λ as the influence of diversity on the response, based on section 5.2.2.

Table 5.4 shows results of the tests of within-subjects for each problem. The table presents the type III sum of squares (SS), the degrees of freedom (DF), the mean squares (MS), the F test statistics (F) and the eta-squared (Eta). Interactions between 2 factors are represented by factor1*factor2. P-values less than 0.01 represent rejection of the null hypothesis that the average response is statistically equal at all the levels of the corresponding factors, considering a significance level of 1%. Only factors/interactions involving λ or with effect size larger than 0.10 are shown in order to facilitate reading. The p-values for these were always less than 0.001, except for λ*Sev*Sp for SineH, in which it was 0.009.

The interactions between time and severity (usually not reported in the table) have frequently medium (eta-squared between 0.023 and 0.080) and in two cases considerably large effect size (about 0.19), whereas interactions between speed and other factors are usually small (always less than 0.005), showing the importance of the drifts categorisation proposed in section 4.1, which distinguishes severity and speed.

As we can observe in table 5.4, not only time, but also λ usually has large effect size. That

Table 5.4: ANOVA – Test of Within-Subjects Effects: Factors/interactions which involve λ or have eta-squared higher than 0.10, in order of effect size for each problem. The p-values were always less than 0.001, except for λ*Sev*Sp for SineH, in which it was 0.009.

Problem	Factor/Interaction	SS	DF	MS	F	Eta
Circle	λ	95.592	2.594	36.853	7130.564	0.559
	Time	31.029	2.771	11.198	10991.842	0.181
	λ * Time	22.968	8.075	2.844	1402.398	0.134
	λ * Sev	5.743	5.188	1.107	214.194	0.034
	λ * Time * Sev	2.138	16.149	0.132	65.266	0.012
	λ * Sp	0.290	5.188	0.056	10.814	0.002
	λ * Time * Sp	0.230	16.149	0.014	7.012	0.001
SineV	λ	114.271	1.773	64.437	6800.644	0.527
	Time	46.030	2.342	19.655	16944.439	0.212
	λ * Time	21.080	5.452	3.866	1191.539	0.097
	λ * Sev	7.874	3.547	2.220	234.308	0.036
	λ * Time * Sev	3.657	10.905	0.335	103.354	0.017
	λ * Sp	0.205	3.547	0.058	6.093	0.001
	λ * Time * Sp	0.312	10.905	0.029	8.818	0.001
SineH	Time	119.824	2.596	46.155	40623.548	0.464
	λ	44.322	3.565	12.433	3477.104	0.172
	λ * Time	41.646	7.825	5.322	2043.490	0.161
	λ * Sev	14.842	7.130	2.082	582.171	0.057
	λ * Time * Sev	6.061	15.651	0.387	148.705	0.023
	λ * Sp	0.241	7.130	0.034	9.470	0.001
	λ * Time * Sp	0.180	15.651	0.012	4.424	0.001
	λ * Sev * Sp	0.108	14.259	0.008	2.126	0.000
Line	λ	119.564	1.697	70.468	6709.612	0.564
	Time	39.277	2.111	18.602	15127.604	0.185
	λ * Time	20.044	4.700	4.265	1150.406	0.095
	λ * Sev	8.069	3.393	2.378	226.417	0.038
	λ * Time * Sev	3.104	9.400	0.330	89.068	0.015
	λ * Time * Sp	0.146	9.400	0.016	4.189	0.001
Plane	Time	39.277	2.111	18.602	15127.604	0.385
	Time * Sev	78.368	3.575	21.921	2587.570	0.190
	λ	58.610	3.846	15.240	933.808	0.142
	λ * Sev	36.000	7.692	4.680	286.784	0.087
	λ * Time	23.246	6.241	3.725	241.783	0.056
	λ * Time * Sev	7.368	12.482	0.590	38.318	0.018
	λ * Sp	0.957	7.692	0.124	7.622	0.002
	λ * Time * Sp	0.942	12.482	0.075	4.899	0.002
Bool	Time	198.230	2.047	96.839	23125.086	0.594
	Time * Sev	64.091	4.094	15.655	3738.360	0.192
	λ	29.341	3.696	7.940	918.070	0.088
	λ * Time	8.172	6.135	1.332	166.485	0.024
	λ * Sev	5.127	7.391	0.694	80.207	0.015
	λ * Time * Sev	2.472	12.270	0.201	25.182	0.007
	λ * Time * Sp	0.637	12.270	0.052	6.493	0.002

means that not only the drift, but also diversity has large impact on the response. Excluding Boolean, the interactions between λ and severity always have medium effect size (eta-squared between 0.034 and 0.087) and the interactions between λ and time always have medium or large effect size (eta-squared between 0.056 and 0.161). So, diversity plays important and probably different roles depending on the severity and time (before, shortly after or longer after drift). The effect size of the interactions between λ and speed is always very small (eta-squared 0.002 or less). So, in the rest of this section, we will check what role diversity plays depending on severity and time.

As there are interactions among λ, time and severity for all the problems, plots of marginal means λ vs severity were generated for each problem and time step analysed. The plots for circle are shown in figure 5.2 as a representative example.

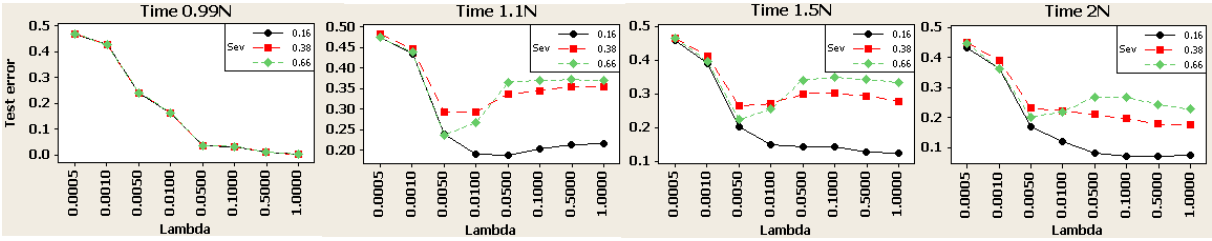


Figure 5.2: Plots of marginal means for the effect of λ*severity*time on the test error for circle.

According to the plots, at the time step $0.99N$ (before drift), the highest λ (lowest diversity) obtained the best responses independent of the severity. The only exception was plane, which is the only problem that contains many irrelevant attributes. In this case, $\lambda = 0.1$ obtained the best responses. For Boolean, $\lambda = 0.05$, 0.1, 0.5 and 1 always obtained response 0 (zero) at this time step.

At the time step $1.1N$ (shortly after the drift), when severity was high, lower λs frequently obtained better responses than $\lambda = 1$. As severity reduced, the best λs started obtaining more similar response to $\lambda = 1$. Besides, the best λ values tended to be higher for the low than for the high severity drifts (higher for 4 problems and the same for 2 problems). So, the higher the severity, the more beneficial is high diversity likely to be shortly after the drift if no additional strategy is adopted to converge to the new concept.

Additional paired T tests (Witten and Frank; 2000) using Bonferroni corrections considering all combinations of severity and diversity with overall significance level of 0.01 (so that a test is considered significant if its associated probability is smaller than 0.01/24) confirm this analysis. They show that, when severity is low, the null hypothesis that the average response is statistically equal using the best average response $\lambda < 1$ and $\lambda = 1$ is rejected for only 2 in 6 problems. However, when severity is medium or high, this number increases to 4.

That is an interesting fact, as one could expect that higher severity would make different diversity ensembles equally bad right after the drift, but, actually, highly diverse ensembles

manage to get better responses when severity is high and it is the lowest severity which makes the behaviour of high and low diversity ensembles more similar. Another interesting fact is that, even though high diversity is more beneficial when severity is high, the response of the best $\lambda < 1$ is always better or similar to $\lambda = 1$ independent of the severity.

At the time step $1.5N$, high diversity is still more important when severity is higher. Besides, additional T tests with Bonferroni corrections reveal that the number of problems in which the best λ obtains a statistically significant different average response from $\lambda = 1$ for low and high severity increases (from 2 to 3 and from 4 to 5, respectively). So, higher diversity is still important at the time step $1.5N$.

At the time step $2N$, the importance of high diversity reduces for all the severities (from 3 to 2, 4 to 2 and 5 to 3 for low, medium and high severities, respectively).

So, we verified that:

1. There are differences between the influence of diversity on the ensemble error before and after the drift. Before the drift, less diversity usually obtained the best results. However, it is a good strategy to maintain high diversity in order to obtain better results on the new concept after the beginning of the drift. After a large number of time steps have passed since the beginning of the drift, high diversity became less important.

2. Drifts with different severities required different amounts of diversity (higher diversity was more important for more severe drifts). However, the effect of diversity on drifts with different speeds was very small.

Besides these points, it is also good to verify which diversity level generally obtains the best average responses, considering all types of drift at the same time. So, it is important to analyse the main effect plots of λ for each time step and check the Q statistics associated to the best λs. The main effect plots for circle are shown in figure 5.3. Although there are frequently 2 exceptions (usually Boolean and plane), we can make the following observations from the main effect plots. The λ values associated to Q statistics higher than 0.85 usually obtained the best average responses before the drift. However, at the time steps $1.1N$ and $1.5N$, Q statistics lower than 0.25 were usually required. At the time step $2N$, Q statistics higher than 0.95 were frequently among the best.

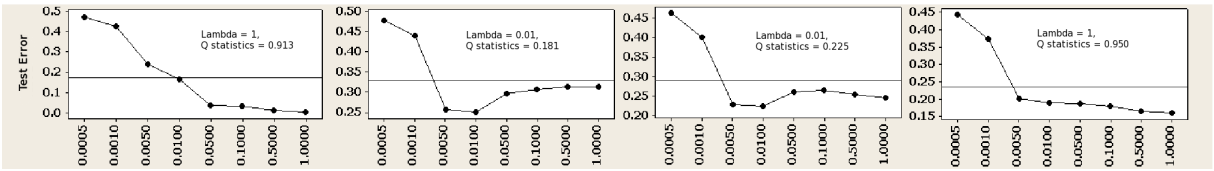


Figure 5.3: Plots of the main effect of λ on the test error at the time steps $0.99N$, $1.1N$, $1.5N$ and $2N$ for circle. Q statistics corresponding to the λs with the lowest test errors are indicated.

5.3.2.2 Sensitivity to Drifts and Adaptation to the New Concept

This section concentrates on checking the points (3) and (4) presented in the beginning of section 5.3: (3) the influence of diversity on the ensemble's sensitivity to drifts and on the adaptation to the new concept considering base learners that learnt the old concept and (4) whether it is possible to exploit diversity to better handle drifts.

First, we shall concentrate on point (3). In section 5.3.2.1, we checked that high diversity ensembles were desirable soon after the drift, for getting similar or better test error on the new concept than low diversity ensembles. However, after a large number of time steps have passed since the beginning of the drift, high diversity became less important. This suggests that, although high diversity ensembles may help to reduce the initial increase in the error soon after the drift (sensitivity to drifts), they are likely not to adapt quickly to the new concept (recovery from drifts).

In order to check if that really happens, the average test error over time was plotted for the λ which obtained the best test error before the drift (time step $0.99N$) and the λ which obtained the best test error soon after the drift (time step $1.1N$). As high diversity helps especially drifts with high severity, the graphs were plotted for the experiments with high severity and high speed. The plots for circle are shown in figure 5.4.

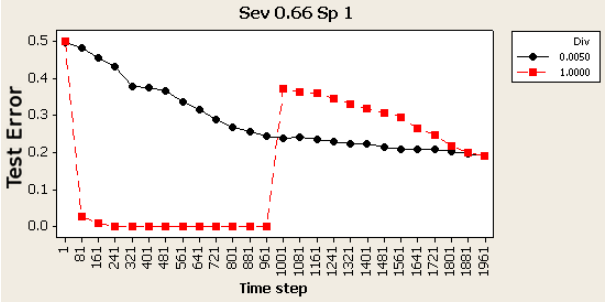


Figure 5.4: Average test error for the highest severity and speed drift for circle. The λ value corresponds to the best test error before the drift ($\lambda = 1$) and the best test error shortly after the drift ($\lambda = 0.005$).

The plots show that the average test error for the higher diversity ensembles was lower shortly after the drift for all problems but Boolean, in which it was similar. However, the average test error for the lower diversity ensembles always decayed faster, so that, in the end of the learning, it usually got similar or lower test error than the higher diversity ensembles. So, high diversity by itself cannot help the ensemble to have a fast recovery from the drift after the initial effect of reducing the increase in the error. Low diversity can get lower error in the absence of drifts and faster recovery from drifts, although it still takes a too long time to attain low error after a drift.

Now, we can address point (4). When designing an approach to handle concept drift, several issues should be considered, as explained in section 1.1.4. One of them is the speed of recovering

(adaptation to the new concept), which is the most addressed in the literature. Another one is the reduction of the initial increase in error which occurs right after a drift (sensitivity to drifts). Other points are obtaining low error in the absence of drifts and efficiently using information from the old concept to learn the new concept whenever it is beneficial. In order to successfully handle concept drift, an approach should address all these issues. So, the answer to point (4) is that diversity by itself can be exploited to better handle concept drift by reducing the test error soon after a drift.

5.3.2.3 Results Using UCI Problems

This section presents the results of the experiments performed with the UCI problems. ANOVA indicates that there is interaction between λ and time for all the problems (null hypothesis of equal means always rejected with p-value less than 0.001). The plots of marginal means are shown in figure 5.5. Only λs corresponding to the best test errors after the drifts and $\lambda = 1$ are shown in order to facilitate reading. We have verified in section 5.3.2.1 that different severities require different amounts of diversity. Here, the best $\lambda < 1$ also varies for different drifts.

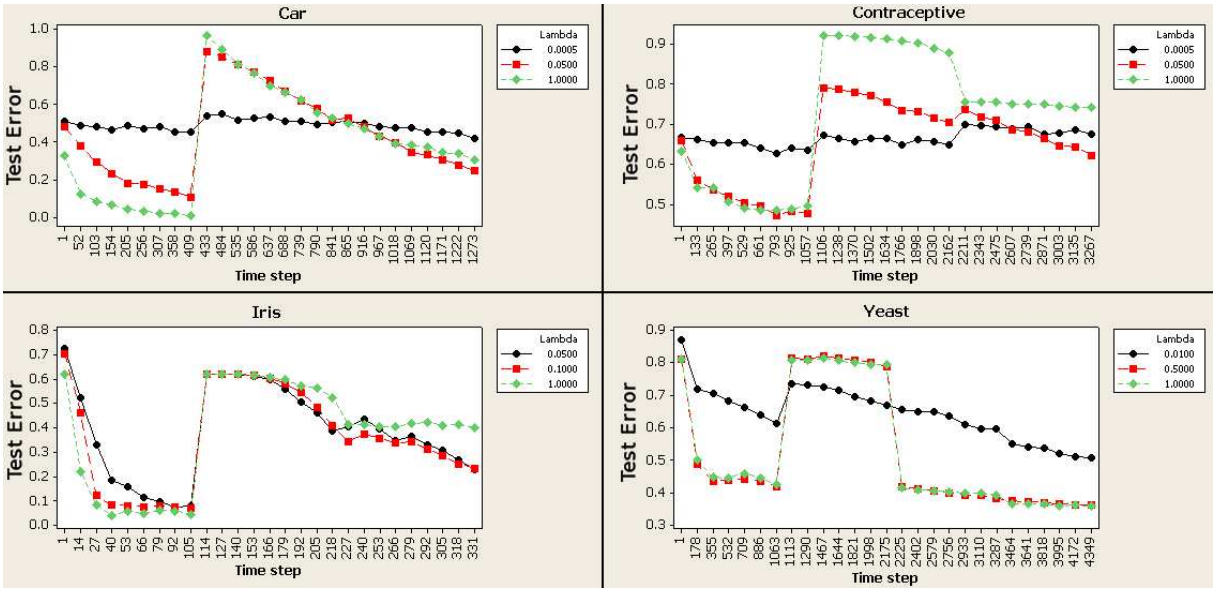


Figure 5.5: Average test error for λs corresponding to the best test errors after the drifts and for $\lambda = 1$, for the UCI problems.

As we can observe, although $\lambda = 1$ obtained good test errors before the drifts, it got worse than a lower λ (higher diversity) after the first drift for all the problems. For iris, there was a delay and a lower λ got lower test error than $\lambda = 1$ only after the time step $1.1N$, but, even so, it got lower test error. We can also observe that very low λs (e.g., 0.0005 for car and contraceptive) did not converge. Not so low λs presented an error decay more similar to $\lambda = 1$. However, they still did not attain low test error on the new concept.

As the second drift presents a partial return to the first concept, higher λs (lower diversity)

than after the first drift presented lower test error. However, except for yeast, λs lower than 1 still presented the lowest errors. A possible explanation for the fact that $\lambda = 1$ obtained good test error after the second and third drifts for yeast is that the new concepts are actually easier than the old concepts. Such situations can happen in real world problems.

The experiments using UCI real world problems with simulated drifts reassure the results presented in the previous sections, showing that high diversity is useful to reduce the initial increase in error when drifts happen. Besides, these experiments suggest that it is worth maintaining ensembles which learnt old concepts, so that they can be used in the case of recurrent drifts.

5.3.3 Discussion

The analysis shows that diversity plays an important role both before and after a concept drift. We can answer points (1) and (2) commented in the beginning of section 5.3 in the following way. The analysis indicates that, before the drift, ensembles with less diversity obtain better test errors. On the other hand, shortly after the drift, the errors of the more diverse ensembles on the new concept are among the best ones. The difference in the test error in comparison to lower diversity ensembles is usually more significant when severity is higher. Even this way, independent of the type of the drift, it is a good strategy to maintain highly diverse ensembles to obtain good test error on the new concept shortly after the drift. At the time step $2N$, less diversity frequently returns to obtain good responses, although some drifts still require higher diversity. The study also shows that the effect of severity on the error on the new concept was higher than the effect of speed, further illustrating the importance of the categorisation proposed in section 4.1.

It is interesting to observe that some parallels could be traced between the use of high/low diversity and exploration/exploitation in reinforcement learning (Berry and Fristedt; 1985). High diversity could be seen as exploration of the space, whereas low diversity could be seen as exploitation.

An important point to be noted here is that ensembles with *lower* diversity are not ensembles with *no* diversity. The value $\lambda = 1$ was the maximum used in the experiments. This is the default value used with online bagging (Oza and Russell; 2001a,b, 2005). So, it approximates the diversity level obtained by offline bagging (Breiman; 1996a), which still benefits the accuracy. As explained in section 5.2, when using $\lambda = 1$, the Q statistic was always less than 1, i.e., there was still diversity. So, we can consider that the high diversity ensembles from this section are ensembles with *very* high diversity. As explained in section 2.1, the accuracy of an ensemble depends not only on the diversity, but also on the accuracy of each ensemble member. When increasing diversity a lot under stable conditions, each ensemble member becomes less accurate, possibly reducing the ensemble accuracy. It is thus reasonable that the high diversity ensembles were less accurate than the lower diversity ensembles before the drift.

The experiments also showed that diversity by itself could mainly help to reduce the initial increase in the error caused by a drift (points (3) and (4) from the beginning of section 5.3), even though it was not able to help convergence to the new concept. We will see in section 5.4 that it is also possible to use diversity to aid the learning of the new concept by adopting the strategy of encouraging less diversity in a high diversity ensemble once the drift begins.

The present section also suggests that the difficulty of the concept learnt before a drift might also influence the learning of the new concept. A concept easily learnt might be more difficult to be forgotten, considerably increasing the error obtained after a drift depending on the difficulty of the new concept. So, further studies should be done to investigate that. Another hypothesis to be investigated is that higher diversity may help learning when there are many irrelevant attributes, as the only database in which a λ lower than 1 ($\lambda = 0.1$) provided the best test error before the drift was Plane, which contains many irrelevant attributes.

5.4 The Influence of Diversity on the Prequential Accuracy

The analysis presented in section 5.3 emphasizes the impact of diversity on either the old or the new concept. It does not use any special feature for dealing with drifts in order to analyse the impact of diversity by itself.

In the present section, an analysis of diversity using the average prequential accuracy (Dawid and Vovk; 1999) is done. The average is calculated in an online way, using the prediction given to each example before its learning. So, it considers the instability of the correctness of the predictions during the drifting time, instead of verifying only the impact on either concept. The analysis also uses additional strategies (such as resetting the ensemble) together with diversity in order to encourage convergence to the new concept. Among them, the best strategy for each type of drift is identified. This section extends the answer to the research question presented in section 1.2.2 and answers the research question presented in section 1.2.3: "Can we use information from the old concept to better deal with the new concept? How?" In order to do so, the following points are analysed:

1. Online learning often operates in the scenario explained by Littlestone and Warmuth (1994) and further adopted in many works, such as Kasabov (2003), Fern and Givan (2003), Gama et al. (2004) and Baena-García et al. (2006):

   > Learning proceeds in a sequence of trials. In each trial the algorithm receives an *instance* from some fixed *domain* and is to produce a binary *prediction*. At the end of the trial the algorithm receives a binary *label*, which can be viewed as the correct prediction for the instance.

   Several real world applications operate in this sort of scenario, such as spam detection, pre-

diction of conditional branches in microprocessors, information filtering, face recognition, etc. So, it is important to analyse the prequential accuracy, considering the prediction done at each trial. Besides, during gradual drifts, the system might be required to make predictions on instances belonging to both the old and new concepts, emphasizing the importance of analysing the prequential accuracy, which considers examples of both concepts at the same time when the drift is gradual. So, how would a low and a high diversity ensemble behave considering the prequential accuracy in the presence of different types of drift?

2. Even though high diversity ensembles may obtain higher accuracy than low diversity ensembles after the beginning of a drift, the study in section 5.3 also reveals that high diversity ensembles are likely to present almost no convergence to the new concept, having slow recovery from drifts. So, is it possible to make a high diversity ensemble trained on the old concept converge to the new concept? How?

3. If it is possible, would that ensemble outperform a new ensemble created from scratch when the drift begins? For which types of drift? If the answer to this question is affirmative, this would be a way to use information from the old concept to aid the learning of the new concept.

The analysis shows that one of the ways to make highly diverse ensembles trained on the old concept converge to the new concept is to start learning the new concept with low diversity. In this way, it is possible to use information learnt from the old concept in order to aid the learning of the new concept. Considering the prequential accuracy, the experiments indicate that these ensembles are usually the best when (1) the drifts do not cause many changes or (2) long after drifts in which both the old and new concepts are kept active for a certain period of time (gradual drifts). In these cases, these ensembles are likely to be even more accurate than new ensembles created from scratch at the beginning of the drift, which is the strategy adopted by many approaches in the literature (Gama et al.; 2004; Baena-García et al.; 2006; Nishida and Yamauchi; 2007b). New low diversity ensembles created from scratch at the beginning of the drift are usually the most accurate when the drift causes very big changes and occurs suddenly. Low diversity ensembles trained on the old concept are usually the most accurate shortly after the beginning of gradual drifts.

This section is further divided as follows: section 5.4.1 presents the experimental design and measures analysed. Section 5.4.2 presents the analysis itself. Section 5.4.3 gives a discussion.

5.4.1 Experimental Design

This study addresses the three points explained in the beginning of section 5.4 by analysing the prequential accuracy of the ensembles presented in table 5.5. The prequential accuracy (Dawid and Vovk; 1999) is the average accuracy obtained by the prediction of each example to be learnt,

before its learning, calculated in an online way. The rule used to obtain the prequential accuracy on time step t is presented in equation 5.1 (Baena-García et al.; 2006):

$$acc(t) = \begin{cases} acc_{ex}(t), & \text{if } t = f \\ acc(t-1) + \frac{acc_{ex}(t) - acc(t-1)}{t-f+1}, & \text{otherwise} \end{cases} \quad (5.1)$$

where acc_{ex} is 0 if the prediction of the current training example ex before its learning is wrong and 1 if it is correct; and f is the first time step used in the calculation.

As shown in table 5.5, before the drift, a new low diversity and a new high diversity ensemble are created from scratch. After the drift begins, the ensembles created before the drift are kept and referred to as *old* ensembles. The old high diversity ensemble starts learning with low diversity, in order to check if it is possible to converge to the new concept. In addition, a new low and a new high diversity ensemble are created from scratch. These ensembles represent different strategies which may be used to deal with drifts. The study identifies which of these strategies are the most accurate for each type of drift.

Table 5.5: Ensembles/Strategies Analized.

Before the drift	After the beginning of the drift	Points addressed
New low diversity →	Old low diversity	1
New high diversity →	Old high diversity now learning with low diversity	1, 2 and 3
	New low diversity	1 and 3
	New high diversity	1

As the main aim of this study is to identify which ensemble/strategy provides the best convergence to a new concept for each type of drift, we need to know exactly what type of drift is present in the data set. So, the study uses the artificial data sets presented in section 4.2.2. In order to analyse the behaviour of the ensembles before and after the beginning of a drift, the prequential accuracy shown in the graphs is reset whenever a drift starts ($f \in \{1, N+1\}$). The learning of each ensemble is repeated 30 times for each data set.

As in section 5.3, the online ensemble learning algorithm used in the experiments is the modified version of online bagging explained in section 5.2. The ensembles were composed of 25 ITI online decision trees (Utgoff et al.; 1997), as in sections 5.2 and 5.3. The parameter λ_l for the *poisson* distribution of the low diversity ensembles is 1, which is the value used for the original online bagging (Oza and Russell; 2001a). The λ_h values for the high diversity ensembles were chosen in the following way:

1. Perform 5 preliminary executions using λ_h = 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, giving a total of 35 executions for each data set.

2. Determine the prequential accuracy obtained by each execution at the time step $1.1N$. This time step represents one of the moments in which high diversity ensembles are more likely to outperform low diversity ensembles, according to section 5.3.

3. Calculate the main effect of λ_h on the average prequential accuracy, considering all the data sets for each particular problem at the same time. For example, the main effect for the circle problem is the average among the prequential accuracies obtained by the 5 executions using all the 9 circle data sets.

4. For each problem, choose the λ_h which obtained the best main effect. These values are 0.05 for circle, sineH and plane, and 0.005 for line and sineV. For Boolean, both $\lambda_h = 0.1$ and $\lambda_h = 0.5$ obtained the same main effect. So, $\lambda_h = 0.1$ was chosen, as it obtained the best main effect at $1.5N$ and $2N$.

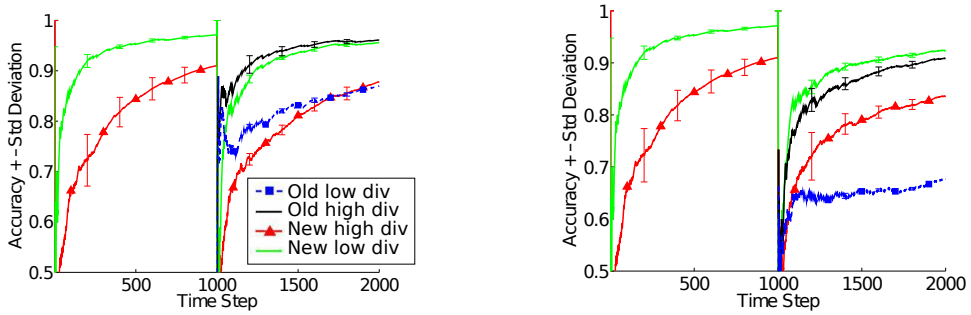
5.4.2 Analysis

Figure 5.6 shows a representative example of the average prequential accuracy obtained for the circle problem for low and high severities and speeds. In order to address the first point from the beginning of section 5.4, the prequential accuracy is analysed **after** the beginning of the drift, for each data set. We can observe that different ensembles obtained the best prequential accuracy depending on the type of drift.

For drifts with low severity and high speed (e.g., figure 5.6(a)), the best accuracy after the beginning of the drift was usually obtained by the old high diversity ensemble. For the Boolean problem, the old low diversity got similar accuracy to the old high diversity ensemble. So, in general, **it is a good strategy to use the old high diversity ensemble for this type of drift**.

An intuitive explanation for the reason why old high diversity ensembles are helpful for low severity drifts is that, even though the new concept is not the same as the old concept, it is similar to the old concept. When a high level of diversity is enforced, the base learners are forced to classify the training examples very differently from each other. So, the ensemble learns a certain concept only partly, being able to converge to the new concept by learning it with low diversity. At the same time, as the old concept was partly learnt, the old high diversity ensemble can use information learnt from the old concept to aid the learning of the new concept. An old low diversity ensemble would provide information from the old concept, but would have problems to converge to the new concept, as shown in section 5.3.

For drifts with high severity and high speed (e.g., figure 5.6(b)), the new low diversity ensemble usually obtained the best accuracy, even though that accuracy was similar to the old high diversity ensemble's in half of the cases (Boolean, sineV and line). For the Boolean problem, the old high diversity, old low diversity and new low diversity obtained similar accuracy. So, in



(a) Low Severity, High Speed

(b) High Severity, High Speed

(c) Low Severity, Low Speed

(d) High Severity, Low Speed

Figure 5.6: Average prequential accuracy (equation 5.1) of the four ensembles analysed for the circle problem considering 30 runs using "perfect" drift detections. The accuracy is reset when the drift starts ($f \in \{1, 1001\}$). The new ensembles are created from scratch at the time steps 1 and 1001. The old ensembles correspond to the new ensembles before the beginning of the drift.

general, **it is a good strategy to use the new low diversity ensemble for this type of drift**.

The reason for that is that, when a drift has high severity and high speed, it causes big changes very suddenly. In this case, the new concept has almost no similarities to the old concept. So, an ensemble which learnt the old concept either partly or fully will not be so helpful (and could be even harmful) for the accuracy on the new concept. A new ensemble learning from scratch is thus the best option.

For drifts with medium severity and high speed, the behaviour of the ensembles was similar to when the severity is high for sineH, circle, plane and line, although the difference between the old high diversity and the new low diversity ensemble tended to be smaller for sineH, circle and plane. The behaviour for Boolean and sineV tended to be more similar to when severity is low. So, drifts with medium severity sometimes had similar behaviour to low severity and sometimes to high severity drifts when the speed is high.

For drifts with low speed (e.g., figures 5.6(c) and 5.6(d)), either the old low or both the old ensembles presented the best accuracy in the beginning of the drift, independent of the severity. So, considering only shortly after the beginning of a drift, the best strategy is to use the old low diversity ensemble for slow speed drifts. Longer after the drift, either the old high or both the old

high and the new low diversity ensembles usually obtained the best accuracies. So, considering only longer after the drift, the best strategy is to use the old high diversity ensemble. If we consider both shortly and longer after the drift, **it is a good strategy to use the old high diversity ensemble**, as it is the most likely to have good accuracy during the whole period after the beginning of the drift.

For drifts with medium speed, the behaviour was similar to low speed, although the period of time in which the old ensembles had the best accuracies was reduced and the old low diversity ensemble rarely had the best accuracy by itself shortly after the beginning of the drift (it usually obtained similar accuracy to the old high diversity ensemble). When the severity was high, there were two cases (sineH and plane) in which the best accuracy was obtained by the new low diversity ensemble longer after the drift. This behaviour approximates the behaviour obtained when the severity is high and the speed is high.

5.4.3 Discussion

The analysis presented in section 5.4.2 directly addresses the first point mentioned in the beginning of section 5.4 and also allows us to answer the second and third points:

(2) Ensembles which learnt an old concept using high diversity can converge to a new concept if they start learning the new concept with low diversity. This is a way to use information from the old concept to aid the learning of the new concept, as it can obtain better accuracy than a new ensemble created from scratch in several cases.

(3) The experiments indicate that, when the drifts have low severity and high speed or longer after drifts with low or medium speed, the old high diversity ensembles learning with low diversity are the most accurate in most of the cases. Besides, shortly after the beginning of drifts with low or medium speed, they are also more accurate than the new ensembles and frequently achieve similar accuracy to the old low diversity ensembles. Even when the drift has medium or high severity and high speed, the old high diversity ensembles sometimes obtain similar accuracy to the new low diversity ensembles. So, in fact, it is a good strategy to use the old high diversity ensembles for most types of drift.

From the above, we can also conclude that the strategy of resetting the learning system as soon as a drift is detected, which is adopted by many approaches, such as Gama et al. (2004); Baena-García et al. (2006); Nishida and Yamauchi (2007b), is not always ideal, as an ensemble which learnt the old concept can be helpful depending on the type of drift.

The study presented in section 5.4 differs from the study presented in section 5.3 in an important point: it analyses strategies to converge to the new concept. In section 5.3, we saw that diversity by itself could help to reduce the initial drop in accuracy on the new concept caused by a drift, but not to improve convergence to the new concept. High diversity by itself

was considered helpful independent of the type of drift, even though it was more helpful for drifts with higher severity.

In the present section, we can see that the strategy of encouraging less diversity in an old high diversity ensemble helped mainly drifts with low severity or low speed. When the drifts had high severity and speed, comparably to section 5.3, ensembles which learnt the old concept with high diversity also presented better behaviour than ensembles which learnt the old concept with low diversity. However, the strategy of creating a new low diversity ensemble upon start of drift was more appropriate for this case. Nevertheless, in real world situations in which drift detections have delays, ensembles trained on the old concept with high diversity could still be helpful for this type of drift because they are likely to be better on the new concept than ensembles trained on the old concept with low diversity, as explained in section 5.3.

5.5 Summary

The contributions of this chapter are:

- A better understanding of when, how and why ensembles of learning machines can help to handle concept drift in online learning, through a diversity study in online changing environments (sections 5.3 and 5.4). This is an answer to research question 1.2.2: "when, how and why can ensembles be helpful for dealing with drifts?"

  Although ensembles have been used to handle concept drift, the literature did not contain any deep study of why they can be helpful for that and which of their features can contribute or not to deal with concept drift. One of the ensembles' features which can make them useful for dealing with concept drift is diversity.

  Section 5.3 first analyses the influence of diversity on the learning of old/new concepts with no additional strategy to encourage convergence to the new concept. It shows that, even though low diversity ensembles are likely to be more accurate on the old concept before the drift, high diversity ensembles can reduce the drop in accuracy on the new concept right after the drift. The analysis also indicates that the higher the severity of the drift the most important the use of higher diversity in order to reduce the drop in accuracy.

  However, additional strategies are necessary in order to encourage convergence to the new concept. So, in section 5.4, four different learning strategies are analysed: (1) to simply train on the new concept an old ensemble trained with low diversity on the old concept; (2) to use an ensemble trained with high diversity on the old concept, but with low diversity on the new concept; (3) to create a new ensemble to learn the new concept from scratch with low diversity; and (4) to create a new ensemble to learn the new concept from scratch with high diversity. Besides, instead of analysing the accuracy on either the old or new concepts, the prequential accuracy is verified. In this way, the uncertainty of the correctness of the

predictions during the drifting time is also considered.

The analysis reveals that the strategy (1), which is equivalent to the low diversity ensemble analysed in section 5.3, can be beneficial during the instability period existing during the drifting time. That happens because some of the instances to be predicted still reflect the old concept. The strategy (2) allows us to both reduce the initial drop in accuracy caused by a drift and encourage convergence to the new concept. It is a way to use information from the old concept to aid the learning of the new concept. The analysis indicates that this strategy is useful mainly for drifts with low severity or speed, as the old concept is useful in these cases. When the drift causes big changes very quickly (high severity and speed), information from the old concept is not useful and the most appropriate strategy is (3). The strategy (4) is not helpful for making predictions on the current concept. However, if a drift is not detected early, this strategy can reduce the initial drop in accuracy before the drift detection, as it would be equivalent to the high diversity ensemble from section 5.3.

- Knowledge about how to use information from the old concept in order to aid the learning of the new concept (section 5.4). This is an answer to research question 1.2.3: "can we use information from the old concept to better deal with the new concept? How?"

  Following the intuition, if the drifts do not cause so much changes in the concept, it should be possible to use information from the old concept to aid the learning of the new concept. Nevertheless, to the best of our knowledge, there is no approach in the literature which does so. They at most allow old classifiers to be maintained and simply trained on the new concept. Section 5.3 shows that simply training an old classifier on the new concept does not provide good convergence to the new concept.

  As explained in the previous bullet point, it is possible to use information from the old concept to aid the learning of the new concept by enforcing low diversity in a high diversity ensemble once the drift begins. The high diversity allows the old concept to be learnt only partially. In this way, some information learnt from the old concept can be used to provide better convergence to the new concept, instead of hindering it. The analysis indicates that this is useful especially for slow drifts or drifts with low severity.

- A technique to directly encourage more or less diversity in online bagging ensembles (section 5.2).

  This chapter introduces a simple technique which allows us to consistently "tune" diversity in online bagging ensembles. This is helpful for making better use of diversity under different drift conditions.

The study in this chapter reveals that the potential of ensembles to handle concept drift is not fully exploited in the literature. Besides, it shows how to use diversity in combination with other strategies in order to obtain higher accuracy for each type of drift. Based on that, chapter 6 proposes a new approach to handle concept drift. The approach is accurate both in

the presence and absence of drifts in comparison to other approaches in the literature, further confirming the value of the study shown in the present chapter.

Chapter 6

Diversity for Dealing with Drifts (DDD)

Chapter 5 shows how diversity and other strategies can be used to help dealing with each type of concept drift. However, in a real world situation, we do not know what sort of drift is present in the data stream. So, we are still left with the research question presented in section 1.2.4: "is it possible to create an approach more robust and accurate considering different types of drift and, at the same time, achieve good accuracy in the absence of drifts?"

Two different directions can be taken to answer this question: (1) to design a drift detection method which is able to identify the type of drift and then use the most appropriate strategy according to section 5.4 or (2) to propose an approach which is more robust to several types of drift without necessarily identifying the type of drift. The thesis concentrates on the latter.

A new approach called Diversity for Dealing with Drifts (DDD) is proposed in the present chapter, providing an answer to the research question presented in section 1.2.4. The approach is based on the diversity studies from sections 5.3 and 5.4. It aims at exploiting diversity to handle drifts, being more robust in the presence of false alarms (false positive drift detections) and having faster recovery from drifts than other approaches in the literature. The following points are analysed in order to validate DDD:

1. Whether it has good accuracy in the presence of concept drift (small drop in accuracy in the beginning of the drift and fast recovery from the drift).
2. Whether it has good accuracy in the absence of drifts.
3. The types of drift for which it works better.
4. Its robustness in the presence of false alarms.
5. The explanation of its behaviour.

The analysis using artificial and real world data shows that DDD is likely to perform at least similarly and in many cases better than two other approaches from the concept drift literature both under several drift conditions and in the absence of drifts. This behaviour is achieved thanks to the successful combination of the four different strategies explained in section 5.4.

This chapter is further divided as follows: section 6.1 describes DDD. Sections 6.2 and 6.3 explain the experimental design and analysis done to validate DDD, considering the five points mentioned above. Section 6.4 presents a summary and discussion.

6.1 DDD's Description

DDD (algorithm 6.5) operates in 2 modes: prior to drift detection and after drift detection. Figure 6.1 presents a diagram of DDD's working. The approach uses a drift detection method, instead of treating drifts implicitly, in order to perform immediate treatment of drifts once they are detected. So, if the parameters of the drift detection method are tuned to detect drifts the earliest possible and the approach is designed to be robust to false alarms, we can obtain fast adaptation to new concepts.

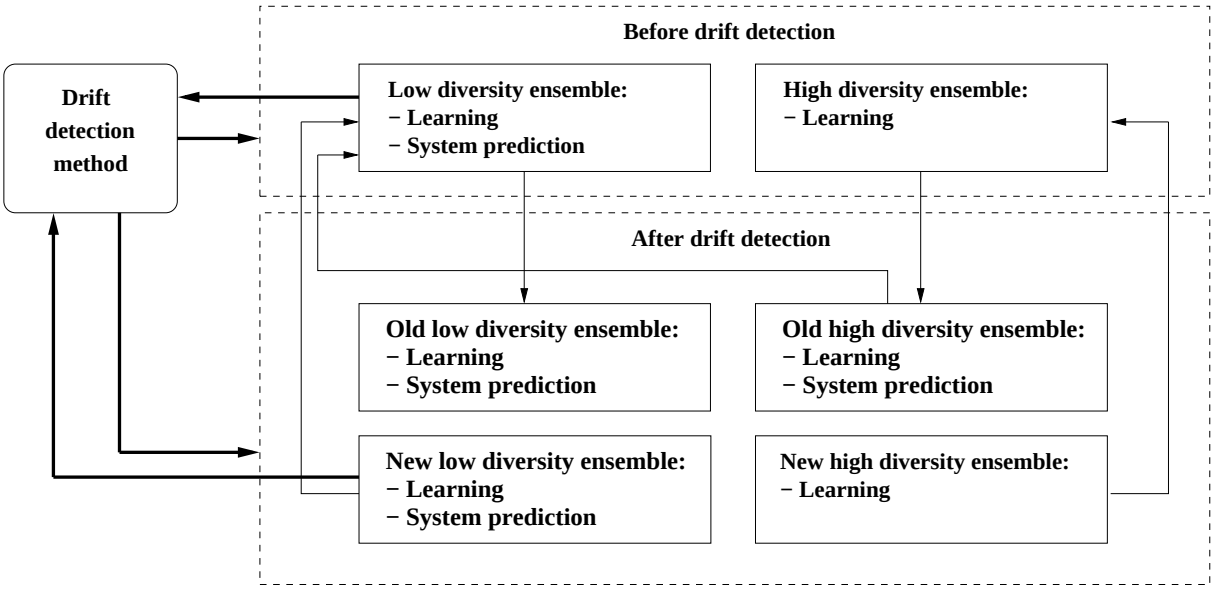


Figure 6.1: Diagram of DDD's overall working. The thick arrows represent flow of information between modules. The thin arrows represent which ensembles in the mode prior to drift detection can become ensembles in the mode after drift detection and vice-versa.

Algorithm 6.5 DDD

Inputs:

- multiplier constant W for the weight of the old low diversity ensemble;
- online ensemble learning algorithm $EnsembleLearning$;
- parameters for ensemble learning with low diversity p_l and high diversity p_h;
- drift detection method $DetectDrift$;
- parameters for drift detection method p_d;
- data stream D;

1: mode ← before drift
2: h_{nl} ← new ensemble /* new low diversity ensemble */
3: h_{nh} ← new ensemble /* new high diversity ensemble */
4: $h_{ol} \leftarrow h_{oh} \leftarrow$ null /* old low and high diversity ensembles */
5: $acc_{ol} \leftarrow acc_{oh} \leftarrow acc_{nl} \leftarrow acc_{nh} \leftarrow 0$ /* accuracies */
6: $std_{ol} \leftarrow std_{oh} \leftarrow std_{nl} \leftarrow std_{nh} \leftarrow 0$ /* standard deviations */
7: **while** true **do**
8: $d \leftarrow$ next example from D
9: **if** mode == before drift **then**
10: prediction ← $h_{nl}(d)$
11: **else**
12: $sum_{acc} \leftarrow acc_{nl} + acc_{ol} * W + acc_{oh}$
13: $w_{nl} = acc_{nl}/sum_{acc}$
14: $w_{ol} = acc_{ol} * W/sum_{acc}$
15: $w_{oh} = acc_{oh}/sum_{acc}$
16: prediction ← WeightedMajority($h_{nl}(d)$, $h_{ol}(d)$, $h_{oh}(d)$, w_{nl}, w_{ol}, w_{oh})
17: Update(acc_{nl}, std_{nl}, h_{nl}, d)
18: Update(acc_{ol}, std_{ol}, h_{ol}, d)
19: Update(acc_{oh}, std_{oh}, h_{oh}, d)
20: **end if**
21: drift ← DetectDrift(h_{nl}, d, p_d)
22: **if** drift == true **then**
23: **if** mode == before drift OR (mode == after drift AND $acc_{nl} > acc_{oh}$) **then**
24: $h_{ol} \leftarrow h_{nl}$
25: **else**
26: $h_{ol} \leftarrow h_{oh}$
27: **end if**
28: $h_{oh} \leftarrow h_{nh}$

29: $h_{nl} \leftarrow$ new ensemble
30: $h_{nh} \leftarrow$ new ensemble
31: $acc_{ol} \leftarrow acc_{oh} \leftarrow acc_{nl} \leftarrow acc_{nh} \leftarrow 0$
32: $std_{ol} \leftarrow std_{oh} \leftarrow std_{nl} \leftarrow std_{nh} \leftarrow 0$
33: mode $\leftarrow$ after drift
34: **end if**
35: **if** mode == after drift **then**
36: **if** $acc_{nl} > acc_{oh}$ AND $acc_{nl} > acc_{ol}$ **then**
37: mode $\leftarrow$ before drift
38: **else**
39: **if** $acc_{oh} - std_{oh} > acc_{nl} + std_{nl}$ AND $acc_{oh} - std_{oh} > acc_{ol} + std_{ol}$ **then**
40: $h_{nl} \leftarrow h_{oh}$
41: $acc_{nl} \leftarrow acc_{oh}$
42: mode $\leftarrow$ before drift
43: **end if**
44: **end if**
45: **end if**
46: EnsembleLearning(h_{nl}, d, p_l)
47: EnsembleLearning(h_{nh}, d, p_h)
48: **if** mode == after drift **then**
49: EnsembleLearning(h_{ol}, d, p_l)
50: EnsembleLearning(h_{oh}, d, p_l)
51: **end if**
52: **if** mode == before drift **then**
53: **Output** h_{nl}, prediction
54: **else**
55: **Output** $h_{nl}, h_{ol}, h_{oh}, w_{nl}, w_{ol}, w_{oh}$, prediction
56: **end if**
57: **end while**

Before a drift is detected, the learning system is composed of two ensembles: an ensemble with lower diversity (h_{nl}) and an ensemble with higher diversity (h_{nh}). Both ensembles are trained with incoming examples (lines 46 and 47), but only the low diversity ensemble is used for system predictions (line 10). The reason for not using the high diversity ensemble for predictions is that it is likely to be less accurate on the new concept being learnt than the low diversity ensemble (section 5.3).

The high diversity ensemble could be helpful in the case of late drift detections, as discussed in chapter 5. However, if the drift detections are not very late, using only the low diversity ensemble for the predictions may bring more benefit during stable concepts than the high diversity

ensemble would bring before a drift is detected. Designing a careful strategy to benefit more from the high diversity ensemble before drift detection is proposed as future work. Besides, DDD assumes that, if there is no convergence of the underlying distributions to a stable concept, new drift detections will occur, triggering the mode after drift detection. DDD then allows the use of the high diversity ensemble in the form of an old high diversity ensemble, as explained later in this section.

A drift detection method based on monitoring the low diversity ensemble is used (line 21). This method can be any of the methods existing in the literature, for instance, the ones used by the approaches explained in section 2.5.1. After a drift is detected, new low diversity and high diversity ensembles are created (lines 29 and 30). The ensembles corresponding to the low and high diversity ensembles before the drift detection are kept and denominated old low and old high diversity ensembles (lines 24 and 28). The old high diversity ensemble starts to learn with low diversity (line 50) in order to improve its convergence to the new concept, as explained in section 5.4. Maintaining the old ensembles allows not only a better exploitation of diversity and the use of information learnt from the old concept to aid the learning of the new concept, but also helps the approach to be robust to false alarms.

Both the old and the new ensembles perform learning (lines 46-50) and the system predictions are determined by the weighted majority vote of the output of (1) the old high diversity, (2) the new low diversity and (3) the old low diversity ensemble (lines 12-16). The new high diversity ensemble is again not considered because it is likely to have low accuracy on the new concept.

The weights are proportional to the prequential accuracy since the last drift detection until the previous time step (lines 13-15). The weight of the old low diversity ensemble is multiplied by a constant W (line 15), which allows controlling the trade-off between robustness to false alarms and accuracy in the presence of concept drifts, and all the weights are normalized. It is important to note that weighting the ensembles based on the accuracy after the drift detection is different from the weighting strategy adopted by the approaches in the literature which do not use a drift detection method. Those approaches use weights which are likely to represent more than one concept at the same time and need some time to start reflecting only the new concept.

During the mode after drift detection, the new low diversity ensemble is monitored by the drift detection method (line 21). If two consecutive drift detections happen and there is no shift back to the mode prior to drift detection between them, the old low diversity ensemble after the second drift detection can be either the same as the old high diversity learning with low diversity after the first drift detection or the ensemble corresponding to the new low diversity after the first drift detection, depending on which of them is the most accurate (lines 24 and 26). The reason for that is that, soon after the first drift detection, the new low diversity ensemble may be not accurate enough to become the old low diversity ensemble. This strategy also helps the

approach to be more robust to false alarms.

All the four ensembles are maintained in the system until either the condition in line 36 or the condition in line 39 is satisfied. When one of these conditions is satisfied, the system returns to the mode prior to drift detection. The accuracies when considering whether the old high diversity ensemble is better than the others are reduced/summed to their standard deviations in line 39 to avoid premature return to the mode prior to drift, as this ensemble is more likely to have higher accuracy than the new low diversity very soon after the drift, when the latter learnt just a few examples.

When returning to the mode prior to drift, either the old high diversity or the new low diversity ensemble becomes the low diversity ensemble used in the mode prior to drift detection, depending on which of them is the most accurate (lines 36-44). An additional parameter to control the maximum number of time steps in the mode after drift detection can be used to avoid a too long time maintaining four ensembles and is proposed as future work.

As can be observed, DDD is designed to better exploit the advantages of diversity to deal with concept drift than the other approaches in the literature, by maintaining ensembles with different diversity levels, according to the experiments presented in section 5.4. None of the approaches existing in the literature fully exploits the advantages of diversity for dealing with concept drift in such a way.

Besides, the approaches which use old classifiers as part of an ensemble in the literature, such as Stanley (2003); Kolter and Maloof (2003, 2005), do not adopt any strategy to improve their learning on the new concept. Nevertheless, as it was shown in section 5.3, these classifiers are likely to have low accuracy on the new concept if no additional strategy is adopted to improve their learning. DDD is designed to use information learnt from the old concept in order to aid the learning of the new concept, by training an old high diversity ensemble with low diversity on the new concept. Such a strategy has shown to be successful in section 5.4.

Moreover, the approach is designed to be more robust to false alarms than approaches which reset the system when a drift is detected (Gama et al.; 2004; Baena-García et al.; 2006; Nishida and Yamauchi; 2007b) and to have faster recovery from drifts than approaches which do not use a drift detection method (Kolter and Maloof; 2003, 2005), as it maintains old ensembles after a drift detection, but takes immediate action to treat the drift when it is detected, instead of having to wait for the weights to start reflecting the new concept.

The approach is not yet prepared to take advantage of recurrent or predictable drifts. The use of memories for dealing with these types of drift is proposed as future work.

A detailed analysis of time and memory occupied by DDD is not straightforward, as it depends on the implementation, base learner and source of diversity. However, it is easy to see that, if we have a sequential implementation, the complexity of each ensemble is $O(f(n))$ and the source of diversity does not influence this complexity, DDD would have, in the worst case,

complexity $O(4f(n)) = O(f(n))$. So, DDD does not increase the complexity class of the system in comparison to a single ensemble.

6.2 Experimental Design

The objective of the experiments is to validate DDD and to assist the explanation of its behaviour, showing that it is an answer to the last research question presented in section 1.2. In order to do so, all the five points explained in the beginning of section 6 are analysed. The artificial data sets from section 4.2.2 are used to explain DDD's behaviour and identify the types of drift for which it works better. Real world data sets are used to further confirm and validate DDD.

When testing a new approach to handle concept drift, besides verifying its behaviour through the use of artificial data sets such as the ones presented in section 4.2.2, it is important to further verify if a good behaviour is attained using real world data sets. As this is a further evaluation in addition to the experiments using artificial data, the fact that it is not possible to know exactly what type of drift is being tested is not a problem.

The real world data sets used in the experiments with DDD are: electricity market (Gama et al.; 2004), KDD Cup 1999 network intrusion detection data (Hettich and Bay; 1999) and PAKDD 2009 credit card data (Neurotech; 2009). They were chosen because they are likely to present different features, as explained in the next paragraphs, allowing us to examine the proposed approach's behaviour under different conditions.

In order to understand better the features of the data sets, the prior probability of class 1 at the time step t is estimated using equation 6.1, similarly to Gao, Fan and Han (2007):

$$P(1)(t) = \frac{\sum_{i=t}^{t+w_{size}-1} y(i)}{w_{size}} , \tag{6.1}$$

where $y(i)$ is the target class (1 or 0) of the training example presented at the time step i, and w_{size} is the size of the window of examples considered for calculating the average.

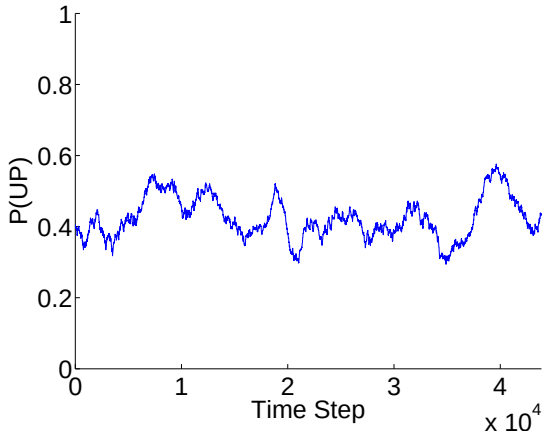
The electricity is a data set from the Australian New South Wales Electricity Market. It was first described in Harries (1999) and has been used in several studies of drift, including online approaches such as DDM, EDDM and DWM (Gama et al.; 2004; Baena-García et al.; 2006; Kolter and Maloof; 2007). In this electricity market, the prices are not fixed and may be affected by demand and supply. Besides, during the time period described in the data, the electricity market was expanded with the inclusion of adjacent areas, causing a dampening of the extreme prices. This data set contains 45,312 examples, dated from May 1996 to December 1998. Each example contains four input attributes (time stamp, day of the week and two electricity

demand values) and the target class, which identifies the change of the price related to a moving average of the last 24 hours. Figure 6.2(a) presents the plot of the estimated prior probability of an increase in price $P(UP)$. As we can see, $P(UP)$ varies a lot during the whole observation period, probably representing several (and possibly continuous) drifts.

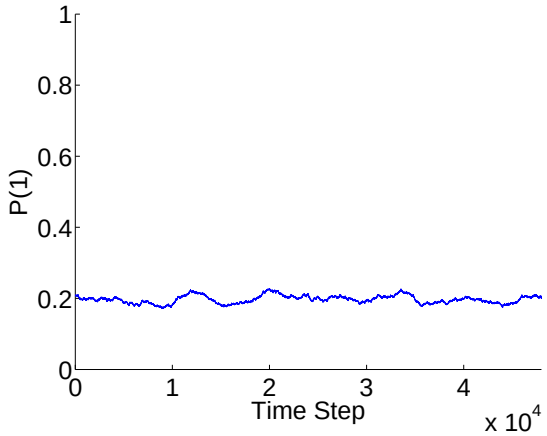
The PAKDD 2009 data set comprises data collected from the private label credit card operation of a major Brazilian retail chain, along stable inflation condition. The experiments use the modelling data, which contains 50,000 examples and correspond to a 1 year period. Each example corresponds to a client and contains 27 input attributes, such as sex, age, marital status, profession, income, etc. The target class identifies if the client is a "good" or a "bad" client. The class "bad" is a minority class and composes around 19.5% of the data. Figure 6.2(b) presents the plot of the estimated prior probability of a client being bad $P(1)$. As we can see, $P(1)$ has almost no variation during the whole learning period. There are only very small fluctuations which may be caused by the sample. It is important to check the behaviour of a drift handling approach under such conditions as well.

The KDD Cup 1999 data set (Hettich and Bay; 1999) contains network intrusion detection data. It has been used for studies of drift, though usually either in incremental learning or presenting only ratios of error rates between the methods compared (Gao, Fan and Han; 2007; Gama and Kosina; 2009; Folino et al.; 2007). The task of an intrusion detection system is to distinguish between intrusions/attacks and normal connections. The data set contains a wide variety of intrusions simulated in a military network environment. During the simulation, the local-area network was operated as if it were a true Air Force environment, but peppered with multiple attacks, so that attack is not a minority class. The data set contains 494,020 examples. Each example corresponds to a connection and contains 41 input attributes, such as the length of the connection, the type of protocol, the network service on the destination, etc. The target class identifies whether the connection is an attack or a normal connection. Figure 6.2(c) presents the plot of the estimated prior probability of having an attack $P(attack)$. As we can see, $P(attack)$ has several jumps from 1 to 0 and vice versa during the first and last third of the learning. So, this data set probably presents several severe and fast drifts which reoccur with a high rate during these periods.

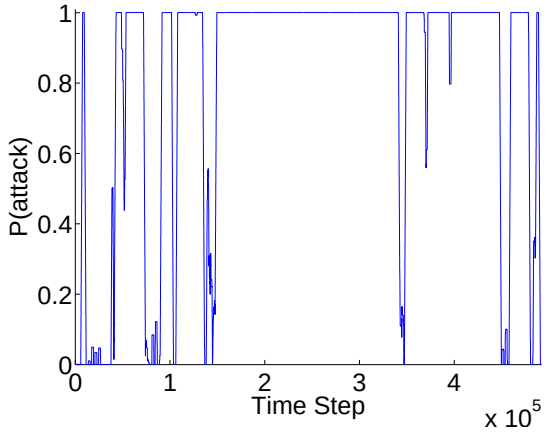
The analysis uses measures such as the weights attributed by DDD to each ensemble, number of drift detections and prequential accuracy (equation 5.1, from section 5.4.1). In some cases, the false positive and negative rates are also analysed. DDD is compared to a low diversity ensemble with no drift handling abilities, an approach which resets the system upon drift detection (EDDM (Baena-García et al.; 2006)) and an approach which does not detect drifts explicitly and maintains old learners unless they become too inaccurate (DWM (Kolter and Maloof; 2003)). The diversity study using "perfect" drift detections presented in section 5.4 and an approach which would always choose to use the same ensemble, i.e., always choose the old high diversity ensemble, or always choose the old low diversity ensemble, etc, are also used in the analysis.



(a) Electricity - Prior probability of an increase in the price $P(UP)$ over time, considering a window corresponding to a month of observations (1440 examples) from the time step analysed.



(b) PAKDD - Prior probability of a client being bad $P(1)$ over time, considering a window of 2000 examples from the time step analysed.



(c) KDD - Prior probability of an attack $P(attack)$ over time, considering a window of 2000 examples from the time step analysed.

Figure 6.2: Histogram of the prior probability over time for the real world problems, estimated according to equation 6.1.

The prequential accuracy is calculated based on the predictions given to the current training example before the example is used for updating any component of the system. It is important to observe that the term *update* here refers not only to the learning of the current training example by the base learners, but also to the changes in weights associated to the base learners (in the case of DWM) and to the ensembles (in the case of DDD). The prequential accuracy was compared both visually, considering the graphs of the average prequential accuracy and standard deviation throughout the learning, and using T student statistical tests (Witten and Frank; 2000) at particular time steps, for the artificial data sets.

The time steps in which the T tests were done are $0.99N$, $1.1N$, $1.5N$ and $2N$. These time steps were chosen in order to analyse the behaviour soon after the drift, fairly longer after the drift and long after the drift, as in section 5.3. Bonferroni corrections considering all the combinations of severity, speed and time step were used. The overall significance level was 0.01, so that the significance level used for each individual comparison was $\alpha = 0.01/(3 * 3 * 4) = 0.000278$. When the difference between two averages is statistically significant and one of the averages is better/worse than the other, we will simply refer to it as being statistically better/worse than the other, to simplify the writing.

The drift detection method used by DDD in the experiments is the same as the one used by EDDM (section 2.5.1), in order to provide a fair comparison. It is important to note that DDD uses only the drift detection method itself. So, differently from EDDM, DDD does not use strategies such as differentiating warning and detection levels and storing training examples. It uses only a drift detection level, determined by the parameter *beta*. There is the possibility that there are other drift detection methods more accurate in the literature. However, the study presented in section 5.4 shows that the old ensembles are particularly useful right after the beginning of the drift. So, a comparison to an approach using a drift detection method which could detect drifts earlier would give more advantages to DDD.

The ensemble learning algorithm used by DDD and EDDM was the (modified) online bagging, as in sections 5.3 and 5.4. The base learners used in the experiments with the artificial data sets were lossless ITI online decision trees (Utgoff et al.; 1997), as in the previous sections. Decision trees were chosen for being one of the most used base learners in the concept drift literature (Gama et al.; 2004; Baena-García et al.; 2006; Oza and Russell; 2005; Kolter and Maloof; 2003; Chu and Zaniolo; 2004; Gao, Fan, Han and Yu; 2007; Wang et al.; 2003; Gao, Fan and Han; 2007; Street and Kim; 2001). The experiments with the real world data sets were repeated using multi-layer perceptions (MLPs) and naive Bayes (NB), which are faster than ITIs for large data sets. NB is also commonly used in the concept drift literature.

The MLPs contained 10 hidden nodes each and were trained using backpropagation with 1 epoch (online backpropagation (Oza and Russell; 2005; Minku and Yao; 2008)), learning rate 0.01 and momentum 0.01. These values were chosen because they provided competitive results in comparison to results published in the literature for Electricity. Both DDD and EDDM used

ensemble size of 25 for the artificial and of 100 for the real world problems. DWM automatically selects the ensemble size, as explained in section 2.5.4.

The sets of values considered for choosing the parameters for the experiment with the artificial data sets and the chosen values themselves are shown in table 6.1. We refer to the parameter β of DWM as ρ in this chapter, not to cause confusion with the parameter β of the drift detection method. The parameters λ_h were chosen to be the same as in section 5.4. The parameter β was chosen so as to provide early drift detections. This parameter was the same for DDD and EDDM.

Table 6.1: Parameters Choice for Artificial Data. $W = 1$, $\lambda_l = 1$ and $\theta = 0.01$ were fixed.

(a) Values for Preliminary Experiments

λ_h (DDD)	β (DDD, EDDM)	α (EDDM)	ρ (DWM)	p (DWM)
{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5}	{0.75, 0.85, 0.95}	{0.96, 0.97, 0.98, 0.99, 1.1, 1.2, 1.3}	{0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.9}	{1, 10, 20}

(b) Chosen Values

Circle	$\lambda_h = 0.05$, $\beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
SineV	$\lambda_h = 0.005$, $\beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
SineH	$\lambda_h = 0.05$, $\beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 10$
Line	$\lambda_h = 0.005$, $\beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 1$
Plane	$\lambda_h = 0.05$, $\beta = 0.95$, $\alpha = 0.99$, $\rho = 0.5$, $p = 5$
Bool	$\lambda_h = 0.1$, $\beta = 0.95$,$\alpha = 0.99$, $\rho = 0.5$, $p = 10$

Preliminary experiments with 5 runs for each data set show that α does not influence much EDDM's accuracy. Even though an increase in α is associated to increases in the total number of time steps in warning level, the system is usually in warning state for very few consecutive time steps before the drift level is triggered, even when very large α is adopted. That number is not enough to significantly affect the accuracy. So, $\alpha = 0.99$ was chosen.

Similarly to λ_h, the parameter p of DWM was chosen considering the best average accuracy at the time step $1.1N$, using the default values of $\rho = 0.5$ and $\theta = 0.01$. The average was calculated using 5 runs for all the data sets of a particular problem at the same time, as for the choice of λ_h. After selecting p, a fine tuning was done for DWM, again based on 5 preliminary runs, by selecting the parameter ρ which provides the best main effect on the prequential accuracy at the time step 1.1N when using the best p. The execution time using $\rho = 0.7$ and 0.9 became extremely high, especially for Plane and Circle. The reason for that is that a combination of a low p with a high ρ causes DWM to include new base learners very frequently, whereas the weights associated to each base learner reduce slowly. So, these values were not considered for

these problems.

The sets of values considered for choosing the parameters for the experiments with the real world data sets and the chosen values themselves are shown in table 6.2. All the parameters were chosen so as to generate the prequential accuracy curves which are overall most accurate, based on 5 runs. The first parameters chosen were β and p, which usually have bigger influence in the accuracy. The 5 runs to choose them used the default values of $\lambda_h = 0.005$ and $\rho = 0.5$. After that, a fine tuning was done by selecting the parameters λ_h and ρ. For electricity, preliminary experiments show that the drift detection method does not provide enough detections. So, instead of using the drift detection method, drift detections were forced at every $FA = \{5, 25, 45\}$ time steps. The only exception was EDDM using NB. In this case, $\beta = 1.15$ provided better results.

Table 6.2: Parameters Choice for Real World Data. $W = 1$, $\lambda_l = 1$ and $\theta = 0.01$ were fixed.

(a) Values for Preliminary Experiments

λ_h (DDD)	β (DDD, EDDM)	α (EDDM)	ρ (DWM)	p (DWM)
{0.005, 0.05, 0.1}	{0.75, 0.95, 1.15}	{0.80, 0.99, 1.20}	{0.3, 0.5, 0.7}	{1, 10, 50}

(b) Chosen Values

	MLP	NB
Electricity	$\lambda_h = 0.005$, $FA = 5$, $\rho = 0.3$, $p = 1$	$\lambda_h = 0.1$, $FA = 45$, $\beta = 1.15$, $\alpha = 1.20$, $\rho = 0.5$, $p = 1$
PAKDD	$\lambda_h = 0.005$, $\beta = 0.75$ $\alpha = 0.80$, $\rho = 0.5$, $p = 50$	$\lambda_h = 0.005$, $\beta = 0.75$ $\alpha = 0.80$, $\rho = 0.5$, $p = 50$
KDD	$\lambda_h = 0.005$, $\beta = 0.95$ $\alpha = 0.99$, $\rho = 0.3$, $p = 1$	$\lambda_h = 0.005$, $DDD\beta = 1.15, EDDM\beta = 0.95$ $\alpha = 0.99$, $\rho = 0.5$, $p = 1$

After choosing the parameters, thirty repetitions were done for each data set and approach.

6.3 Analysis

This section presents the experiments and analysis done to validate DDD. Considering the experimental design explained in section 6.2, the analysis of DDD and its conclusions are based on the following:

1. EDDM always uses new learners created from scratch. Nevertheless, resetting the system upon drift detections is not always the best choice. DWM allows us to use old classifiers,

but does not use any strategy to help the convergence of these classifiers to the new concept. So, it cannot use information from the old concept to learn the new concept faster, as DDD does. At least in the situations in which new learners created from scratch or old learners which attempted to learn the old concept well are not the best choice, DDD will perform better if it manages to identify these situations and adjust the weights of the ensembles properly. This is independent of the base learner.

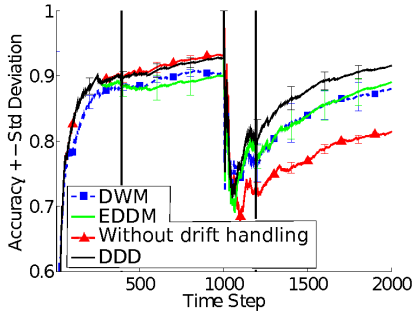
2. The diversity study presented in section 5.4 shows that such situations exist and that an old high diversity ensemble, which is used by DDD, is beneficial for several different types of drift. In section 6.3.1, we verify using artificial data and decision trees that DDD is able to identify such situations and adjust the ensemble weights properly, being accurate in the presence and absence of drifts. The drifts for which DDD works better and the reasons for that are also identified. Section 6.3.2 analyses the number of time steps in which DDD maintains four ensembles. Section 6.3.3 shows that DDD is robust to false alarms and explains the influence of the parameter W. Additional experiments using NB and MLPs on real world problems (section 6.3.4) further confirm the analysis done with the artificial data. Section 6.3.5 comments on the impact of the parameters choice.

6.3.1 Experiments with Artificial Data

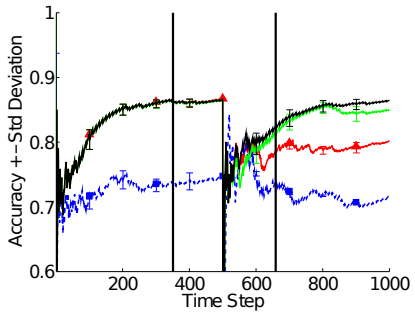
Figure 6.3 shows the prequential accuracy and figure 6.4 shows the weights attributed by DDD for some representative data sets. In this section, we first compare DDD's prequential accuracy to EDDM's. During the first concept, DDD is equivalent to EDDM if there are no false alarms. Otherwise, the experiments show that DDD achieved better accuracy than EDDM. We can observe that through the average number of drift detections during each concept (shown in brackets in figure 6.3); the first vertical black bar (which is used to identify the average time step in which a drift was detected at each concept); and the results of the T tests at the time step $0.99N$ (represented by the first symbol ">", "<" or "=" in each subfigure).

As we can see, DDD obtained better (">") prequential accuracy at the time step $0.99N$ for all databases in which most runs detected drifts during the first concept. These drift detections are all false alarms, as there are no drifts during the first concept. The vertical bars indicate that the prequential accuracies started to become different around the average time step in which a drift was detected. The vertical bars are usually slightly after the beginning of the differences because there were runs detecting more than one drift during the first concept, as indicated by the average number of drift detections in brackets.

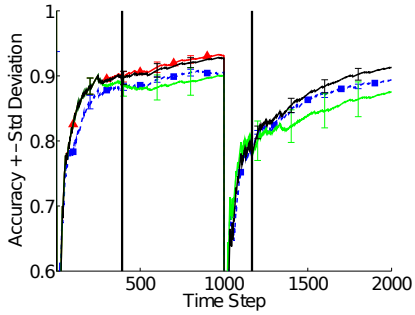
This behaviour is expected, as EDDM resets the system when there is a false alarm, having to re-learn the current concept. DDD, on the other hand, can make use of the old ensembles by increasing their weights. Figure 6.4 shows that indeed DDD increased the weights of the old ensembles when there were false alarms (the average number of drift detections is shown in brackets in figure 6.3).



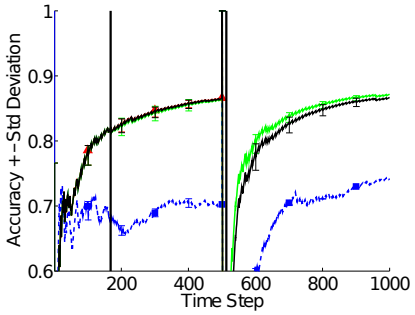
(a) SineH Low Sev High Sp (1.13, 2.23) >=>>



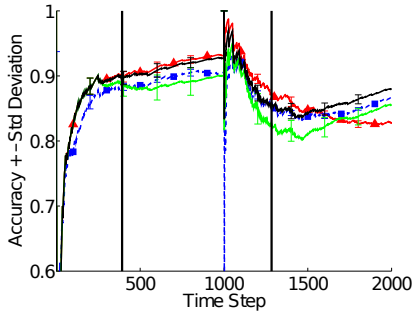
(b) Plane Low Sev High Sp (0.17, 1.47) =>=>



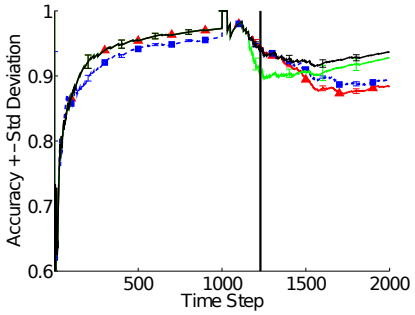
(c) SineH High Sev High Sp (1.13, 2.17) >=>>



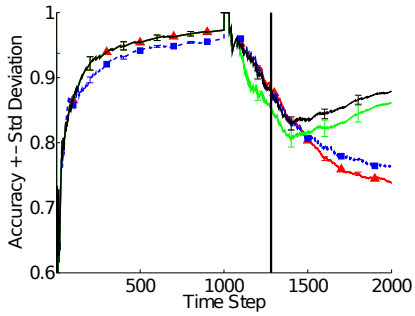
(d) Plane High Sev High Sp (0.10, 1.03) =<<<



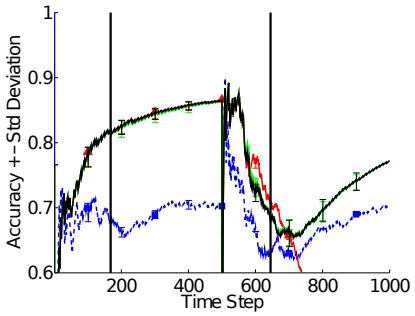
(e) SineH Low Sev Low Sp (1.13, 2.43) >=>>



(f) Circle Low Sev Low Sp (none, 1.26) ==>>

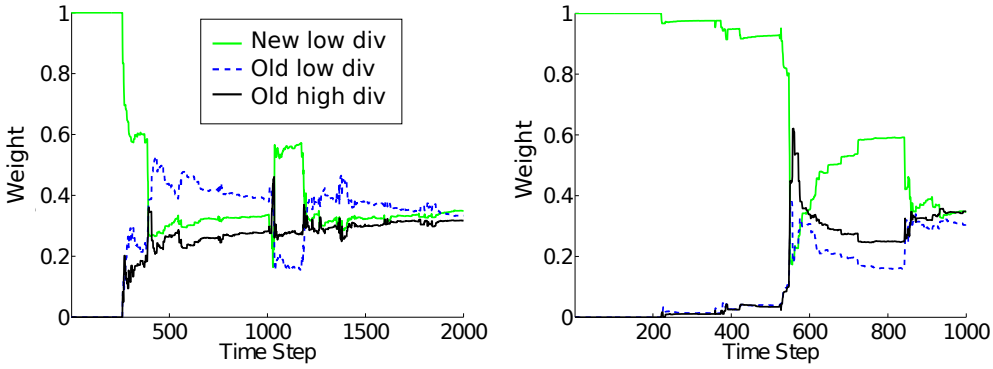


(g) Circle High Sev Low Sp (none, 3.13) =>>>



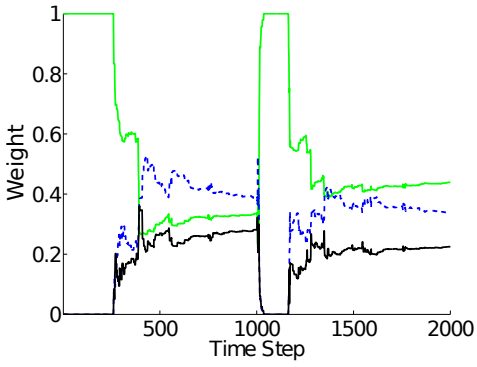
(h) Plane High Sev Low Sp (0.10, 2.09) ====

Figure 6.3: Average prequential accuracy (equation 5.1) of DDD, EDDM, DWM and an ensemble without drift handling considering 30 runs. The accuracy is reset when the drift begins ($f \in \{1, N+1\}$). The vertical black bars represent the average time step in which a drift was detected at each concept. The numbers in brackets are the average numbers of drift detections per concept. The results of the comparisons aided by T tests at the time steps $0.99N$, $1.1N$, $1.5N$ and $2N$ are also shown by four symbols side by side. Symbols ">" mean that DDD attained better accuracy than EDDM, "<" mean worse accuracy and "=" mean similar accuracy.

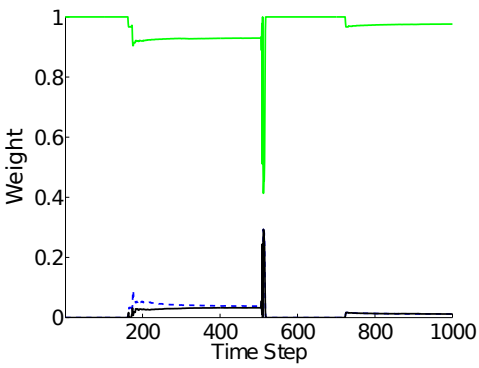


(a) SineH Low Sev High Sp

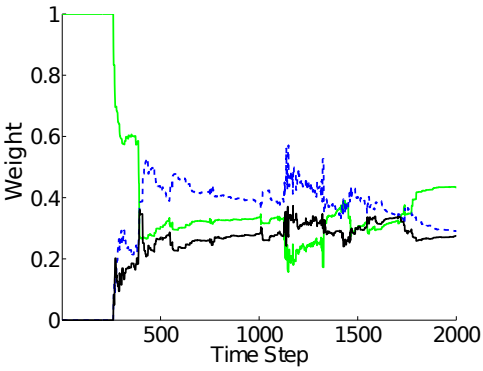
(b) Plane Low Sev High Sp



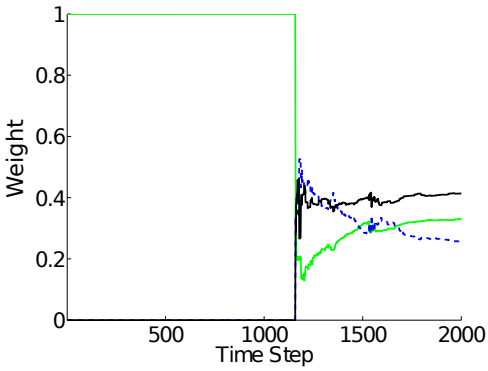
(c) SineH High Sev High Sp



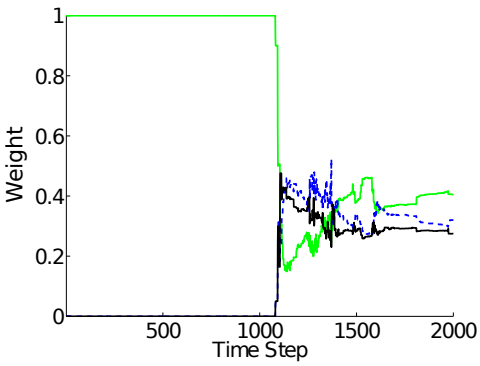
(d) Plane High Sev High Sp



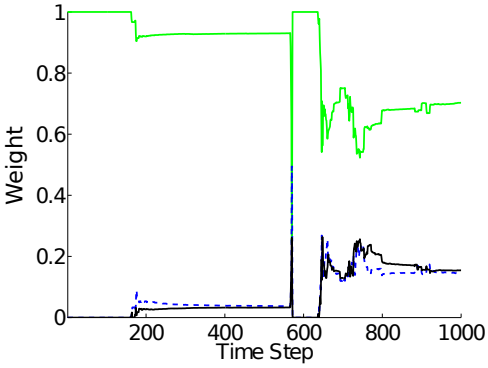
(e) SineH Low Sev Low Sp



(f) Circle Low Sev Low Sp



(g) Circle High Sev Low Sp



(h) Plane High Sev Low Sp

Figure 6.4: Average weight attributed by DDD to each ensemble, considering 30 runs.

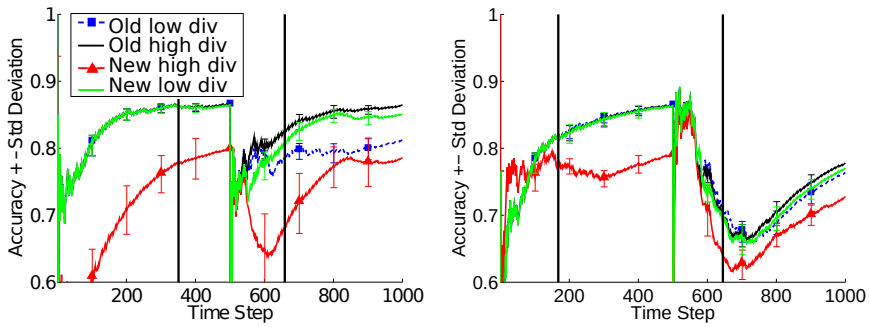
We concentrate now on comparing DDD to EDDM in terms of accuracy after the average time step of the first drift detection done during the second concept. The experiments show that DDD presented better accuracy than EDDM mainly for the drifts known from the diversity study presented in section 5.4 to benefit from the old ensembles (drifts with low severity or low speed). Figures 6.3(a), 6.3(b), 6.3(e), 6.3(f), 6.3(g) show examples of this behaviour.

In the case of low severity and high speed drifts, the study presented in section 5.4 shows that the best ensemble to be used is the old high diversity, especially during the very beginning of the learning of the new concept, when the new low diversity is still inaccurate for having learnt few examples. As shown in figures 6.4(a) and 6.4(b), DDD gave considerable weight to the old high diversity ensemble. Even though it was not always the highest weight, it allowed the approach to get better accuracy than EDDM. When there were false alarms, there were successful sudden rises on the use of the old low diversity ensemble, as shown in the same figures.

However, the non-perfect (late) drift detections sometimes make the use of the old high diversity ensemble less beneficial, causing DDD to get similar, instead of better accuracy than EDDM. Let's observe, for example, figure 6.5(a). This figure shows the average prequential accuracy (equation 5.1) obtained by an approach which always chooses the same ensemble for prediction. For example, an approach which always uses the old high diversity ensemble (black line) for predictions, ignoring the others; or an approach which always uses the new low diversity ensemble (green line), ignoring the others. Even though the study presented in section 5.4 indicates that when there are perfect drift detections, the best ensemble for this type of drift is the old high diversity, the accuracy of an approach which always chooses this ensemble became similar to the new low diversity ensemble's during some moments when the drift detection method was used. DDD obtained similar accuracy to EDDM exactly during these moments (figure 6.3(b)) and just became better again because of the false alarms.

In the case of low speed drifts, the best ensembles to be used according to the study presented in section 5.4 are the old ensembles (especially the old low diversity) soon after the beginning of the drift. DDD managed to attain better accuracy than EDDM (e.g., figures 6.3(e), 6.3(f) and 6.3(g)) because it successfully gave high weight for these ensembles right after the drift detections and kept the weight of the old low diversity ensemble high when there were false alarms, as shown in figures 6.4(e), 6.4(f) and 6.4(g).

The non-perfect (especially the late) drift detections also reduce the usefulness of the old ensembles for the low speed drifts, sometimes making DDD attain similar accuracy to EDDM. An example of this situation is shown by figures 6.3(h), 6.4(h) and 6.5(b). As we can see in figure 6.5(b), the accuracy of an approach which always chooses the old high diversity was similar to an approach which always chooses the new low diversity because of the non-perfect drift detections. For only one problem, when the speed was low and the severity high, the late drift detections made DDD attain worse accuracy than EDDM.



(a) Plane Low Sev High Sp (0.17, 1.47) (b) Plane High Sev Low Sp (0.10, 2.90)

Figure 6.5: Average prequential accuracy (equation 5.1) obtained by an approach which always chooses the same ensemble for prediction, considering 30 runs. The accuracy is reset when the drift begins ($f \in \{1, 501\}$). The vertical black bars represent the average time step in which a drift was detected at each concept. The numbers in brackets are the average numbers of drift detections per concept.

When the drifts present high severity and high speed, the accuracy of DDD was expected to be similar (not better) to EDDM, as the new low diversity is usually the most accurate and EDDM's strategy is equivalent to always choosing this ensemble. However, the experiments show that DDD sometimes presented similar, sometimes worse (figure 6.3(d)) and sometimes better (figure 6.3(c)) accuracy than EDDM.

The reason for the worse accuracy is the inaccuracy of the initial weights given to the new low diversity soon after the drift detection, as this ensemble learnt too few examples. If the initial weights take some time to become more accurate, as shown in figure 6.4(d), DDD needs some time for its prequential accuracy to eventually recover and become similar to EDDM's, as shown by the accuracy's increasing tendency in figure 6.3(d). If the accuracy of the old ensembles decreases very fast in relation to the time taken by the new low diversity ensemble to become more accurate, DDD manages to attain similar accuracy to EDDM since soon after the drift detection. Besides, DDD can attain better accuracy than EDDM even for this type of drift due to the false alarms (figure 6.3(c) and 6.4(c)).

The accuracy of DDD for medium severity or speed drifts was never worse than EDDM's and is explained similarly to the other drifts.

Table 6.3 shows a summary of the win/draw/loss of DDD in comparison to EDDM at the time steps analysed through T tests after the average time step of the first drift detection during the second concept. It is important to note that, when there are false alarms, DDD can get better accuracy than EDDM before this time step.

Considering the total number of win/draw/loss independent of the drift type, DDD obtained better accuracy in 45% of the cases, similar in 48% of the cases and worse in only 7% of the cases. So, it is a good strategy to use DDD when the drift type is unknown, as it is very likely to obtain either better or similar accuracy to EDDM.

Table 6.3: DDD vs. EDDM - Win/Draw/Loss considering the T tests at the time steps after the average time step of the first drift detection during the second concept. The totals independent of severity or speed were 57/62/9 (45%/48%/7%).

Sev	Sp	SineH	Circle	Plane	Boolean	SineV	Line	Total	Total p/sev
Low	High	2/1/0	0/3/0	2/1/0	2/0/0	3/0/0	2/0/0	11/5/0	26/13/0
	Med	2/0/0	3/0/0	0/2/0	1/0/0	0/2/0	0/2/0	6/6/0	67%/33%/0%
	Low	2/0/0	2/0/0	1/1/0	1/0/0	2/0/0	1/1/0	9/2/0	
Med	High	0/3/0	0/3/0	0/3/0	2/0/0	0/3/0	3/0/0	5/12/0	14/30/0
	Med	1/2/0	1/2/0	0/2/0	0/2/0	0/2/0	0/2/0	2/12/0	32%/68%/0%
	Low	2/1/0	2/0/0	0/2/0	0/2/0	1/1/0	2/0/0	7/6/0	
High	High	2/1/0	2/1/0	0/0/3	0/0/3	0/2/1	0/3/0	4/7/7	17/19/9
	Med	3/0/0	2/1/0	0/2/0	0/2/0	1/2/0	0/2/0	6/7/0	38%/42%/20%
	Low	2/1/0	3/0/0	0/2/0	0/0/2	0/2/0	2/0/0	7/5/2	

Considering the totals per severity, we can observe that DDD had more wins (67%) in comparison to draws (33%) or losses (0%) when severity was low. When severity was medium, DDD was similar in most of the cases (68%), being sometimes better (32%) and never worse (0%). When severity was high, DDD was usually similar (42%) or better (38%) than EDDM, but in some cases it was worse (20%). If we consider the totals per speed, the approach had more wins (61%) in comparison to draws (34%) or losses (5%) when speed was low. When the speed was medium, the number of draws was higher (64%, against 36% for wins and 0% for losses). When speed was high, the number of draws and wins was more similar (47% and 39%, respectively), but there were some more losses (14%) than for the other speeds. This behaviour is understandable, as, according to section 5.4, the old ensembles are more helpful when the severity or the speed is low.

The experiments show that DDD attained usually higher accuracy than DWM, both in the presence and absence of drifts (e.g., figures 6.3(a) to 6.3(h)). Before the drift, DDD was almost always better than DWM. Considering the total number of win/draw/loss independent of the drift type for the time steps $1.1N$, $1.5N$ and $2N$, DDD obtained better accuracy than DWM in 59% of the cases, similar in 25% of the cases and worse in 15% of the cases. As we can see in the figures, DDD usually presented faster recovery from drifts.

A probable reason for the lower accuracy of DWM during stable concepts is that it adds new classifiers when it gives misclassifications, independent of how accurate the ensemble is to the current concept. The new classifiers are initially inaccurate and, even though the old classifiers compensate somewhat their misclassifications, the accuracy of the ensemble as a whole is reduced. A probable reason for the lower accuracy of DWM in the presence of drifts is that its weights take some time steps to start reflecting the new concept, causing slow recovery from drifts. So, DWM is usually better than an ensemble without drift handling, but worse than DDD. In a few cases, when drifts that did not affect much the accuracy of old ensembles were detected, DWM obtained better accuracy than DDD.

In a very few occasions, not only DDD, but also EDDM and DWM got worse accuracy than an ensemble without drift handling during a few moments soon after the beginning of gradual drifts (e.g., 6.3(e) and 6.3(h)). That happened because, in the beginning of the drift, ensembles which learnt the old concept are expected to be among the highest accuracies while the old concept is still dominant over the new concept. Nevertheless, as the number of time steps increases and the old concept becomes less dominant, the accuracy of an ensemble without drift handling is highly affected and reduced.

So, overall, DDD got usually similar or better accuracy than EDDM and usually better accuracy than DWM both in the presence and absence of drifts. DDD also usually got better accuracy than an ensemble without drift handling in the presence of drifts and similar accuracy in the absence of drifts.

6.3.2 Time Steps Maintaining Four Ensembles

In this section, the time and memory occupied by DDD are indirectly compared to EDDM and DWM. This is done by considering the number of ensembles maintained in a sequential implementation using λ as the source of diversity and decision trees as the base learners. The high diversity ensembles have faster training and occupy less memory, as they are trained with much less examples (on average, λ times the total number of examples). So, we will compare the number of time steps in which DDD requires four ensembles to the number of time steps in which EDDM requires two ensembles to be maintained.

The experiments presented in section 6.3.1 show that DDD required the maintenance of four ensembles during, on average, 4.11 times more time steps than EDDM required two ensembles. Considering the total number of time steps of the learning, DDD is likely to use, on average, 1.22 times the amount of time and memory used by EDDM. DWM always maintains one ensemble with variable size and this size was, on average, 0.45 times the size of the ensembles maintained by DDD and EDDM. However, DWM is likely to create/delete ensembles with a high rate when the accuracy is not very high, increasing its execution time.

6.3.3 Robustness to False Alarms and the Impact of W

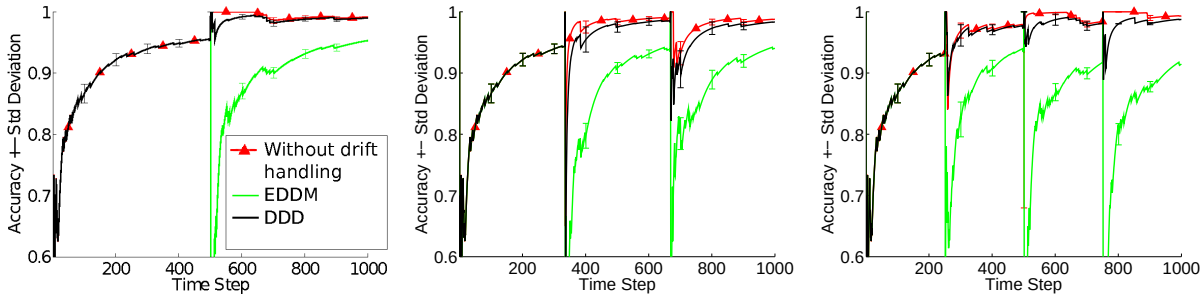
The experiments analysed in section 6.3.1 indicate that DDD is more robust to false alarms (false positive drift detections) than EDDM. However, this issue should still be further investigated to know how well DDD behaves in the presence of false alarms, emphasizing its good behaviour in the absence of drifts. The influence of the parameter W also needs to be further investigated.

In order to do so, experiments using the first concept corresponding to the data sets with low severity and high speed drifts were performed for each artificial problem. Instead of using a drift detection method, a fake drift detection mechanism to simulate false alarms was adopted. The

mechanism triggers a drift detection at every x time steps, where x varied among $N/2$, $N/3$ and $N/4$. So, three scenarios were created, containing from one to three false alarms. The advantage of using this mechanism in addition to the study which detected real false alarms in section 6.3.1 is that all the runs give a false alarm on the same time step, making the analysis more straightforward. Besides, it is possible to study the behaviour considering different frequencies of false alarms.

The parameters were the same as the ones explained in section 6.2, apart from DDD's W. This parameter, which controls the influence of the old low diversity ensemble in a prediction, was varied from 1 (default value) to 3. As increasing W increases the influence of the old low diversity ensemble, the value 3 is expected to make DDD more robust to false alarms, even though it may reduce the accuracy when real drifts occur. DDD is compared to an ensemble without drift handling and to EDDM, which is also based on drift detections. It is important to note that an ensemble without drift handling is expected to be among the best accuracies when there are no drifts.

When using $W = 1$, DDD usually attained much higher accuracy than EDDM after the false alarms, independent of the number of false alarms. Figure 6.6 shows an example for the circle problem. The reason for the higher accuracy is that DDD allows the use of the old ensembles, whereas EDDM resets the system after the detections. This reason is confirmed by the high weights attributed by DDD to the old low diversity ensemble after the false alarms (figure 6.7). This analysis further confirms the good behaviour of DDD to false alarms when using $W = 1$.

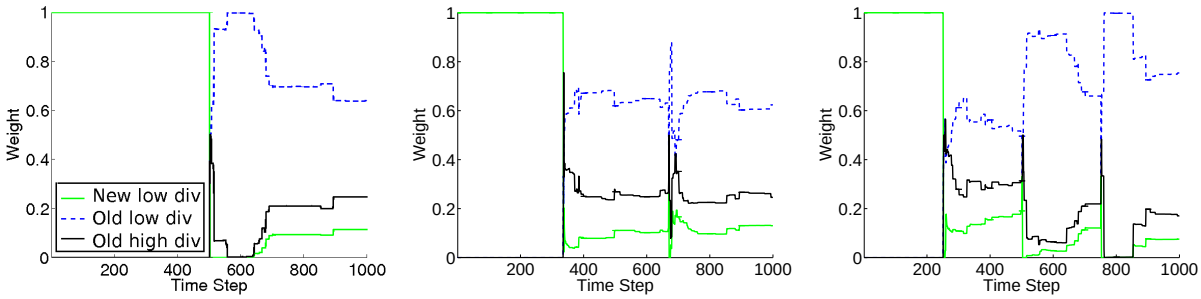


(a) 1 false alarm (b) 2 false alarms (c) 3 false alarms

Figure 6.6: Average prequential accuracy for circle using the default parameter $W = 1$. The accuracy is reset on the time step of the false alarms.

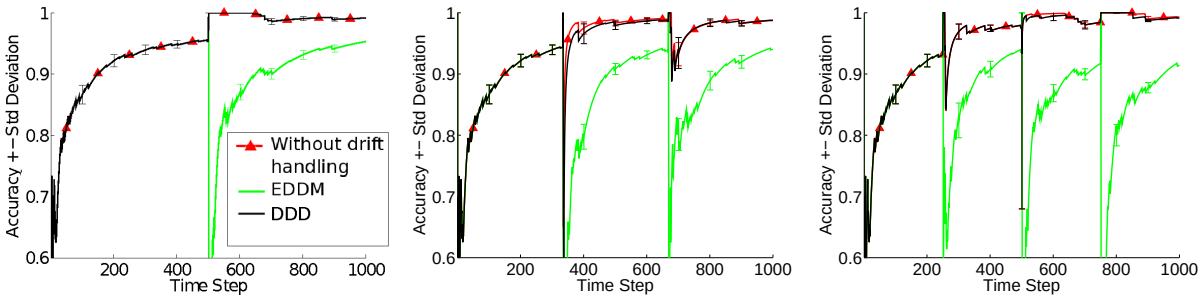
The experiments also confirm that a higher weight ($W = 3$) further increases the robustness of DDD to false alarms. By using this value, when there was only one false alarm, DDD attained very similar accuracy to an ensemble without drift handling. The similarity also increased in the case of two and three false alarms. An example for circle is shown in figures 6.8 and 6.9.

As expected, even though $W = 3$ increases the robustness to false alarms, it makes DDD less accurate when real drifts occur. Additional experiments using all the artificial problems and drifts from section 4.2.2 show that, after the concept drift, $W = 3$ almost always made DDD less accurate than $W = 1$ and, in about half of the data sets, less accurate than EDDM. It is worth



(a) 1 false alarm (b) 2 false alarms (c) 3 false alarms

Figure 6.7: Average weight attributed by DDD to each ensemble, considering 30 runs and using the default parameter $W = 1$.



(a) 1 false alarm (b) 2 false alarms (c) 3 false alarms

Figure 6.8: Average prequential accuracy for circle using $W = 3$. The accuracy is reset on the time step of the false alarms.

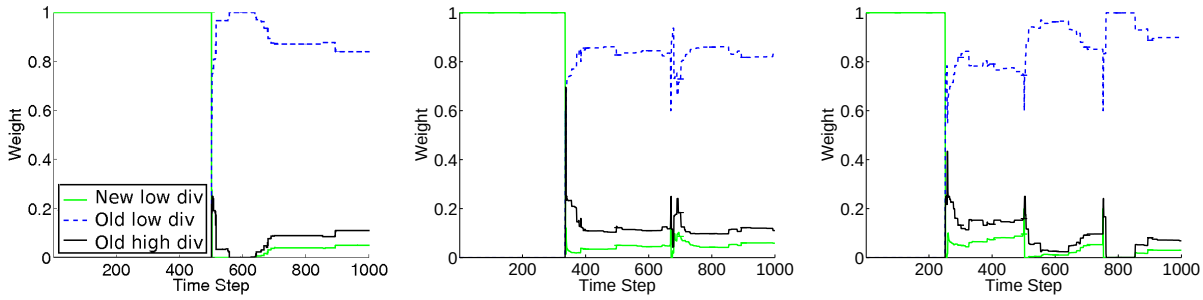
reminding, though, that DDD using $W = 3$ still usually got better accuracy than an ensemble without drift detection when there was a real drift.

So, the parameter W allows tuning the trade-off between robustness to false alarms and accuracy in the presence of real drifts. Higher W ($W = 3$) can make DDD more robust to false alarms and less accurate in the presence of real drifts, but still more accurate than an ensemble without drift handling in the presence of drifts. Lower W ($W = 1$) can make DDD less robust to false alarms, but still with considerably good robustness and more robust than EDDM, besides being more accurate in the presence of real drifts. So, unless we are expecting to have many false alarms and few real drifts, it is a good strategy to use $W = 1$.

6.3.4 Experiments with Real World Data

Each real world data set used in the experiments has different features, as explained in section 6.2. In this way, we can evaluate DDD's performance under different real world situations.

The first data set analysed is electricity. In this data set, the prior probability of an increase in price calculated considering the previous 1440 examples (1 month of observations) varies smoothly during the whole learning period, as shown in figure 6.2(a). These variations



(a) 1 false alarm (b) 2 false alarms (c) 3 false alarms

Figure 6.9: Average weight attributed by DDD to each ensemble, considering 30 runs and using $W = 3$.

possibly represent several continuous drifts, to which DDD is expected to have a good behaviour. Figures 6.10(a) and 6.11(a) show the accuracy obtained for this data set using MLPs and NB, respectively. As we can see, DDD was able to outperform DWM and EDDM in terms of accuracy.

The second data set analysed is PAKDD. In this data set, the probability of a fraudulent customer considering the previous 2000 examples has almost no variation during the whole learning period, as shown in figure 6.2(b). So, this data set is likely to contain no drifts. In this case, DDD is expected to obtain at least similar accuracy to EDDM and DWM. If there are false alarms, DDD is expected to outperform EDDM. The experiments show that all the approaches managed to attain similar accuracy for this problem when using MLPs (figure 6.10(b)). A similar behaviour happened when using NB (figure 6.11(b)). In particular, the drift detection method performed almost no drift detections – average of 0.37 drift detections during the whole learning when using MLPs and of 3.60 when using NB. So, EDDM did not have problems with false alarms. Extra experiments using a parameters setup which causes more false alarms show that DDD was able to maintain the same accuracy as DWM in that condition, whereas EDDM's accuracy reduced.

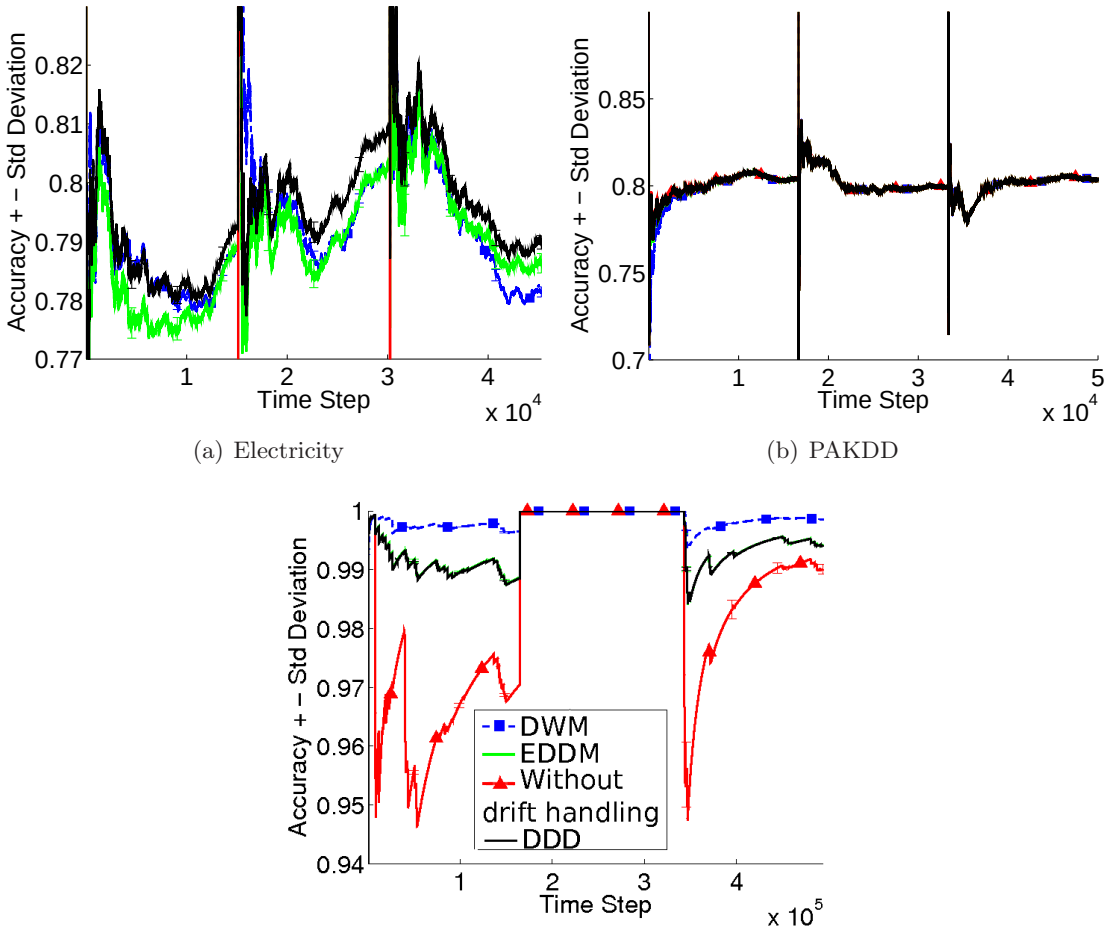
Nevertheless, the class representing a fraudulent customer (class 1) is a minority class. So, it is important to observe the rate of false positives r_{fp} and negatives r_{fn}, which are calculated as follows:

$$r_{fp}(t) = num_{fp}(t)/num_n(t) \text{ and}$$

$$r_{fn}(t) = num_{fn}(t)/num_p(t),$$

where $num_{fp}(t)$ and $num_{fn}(t)$ are the total number of false positives and negatives until the time step t; and $num_n(t)$ and $num_p(t)$ are the total number of examples with true class zero and one until the time step t.

In PAKDD, it is important to obtain a low false negative rate, in order to avoid fraud. When attempting to maximize accuracy, the false positive rate becomes very low, but the false negative rate becomes very high for all the approaches. So, the parameters which obtain the



(a) Electricity

(b) PAKDD

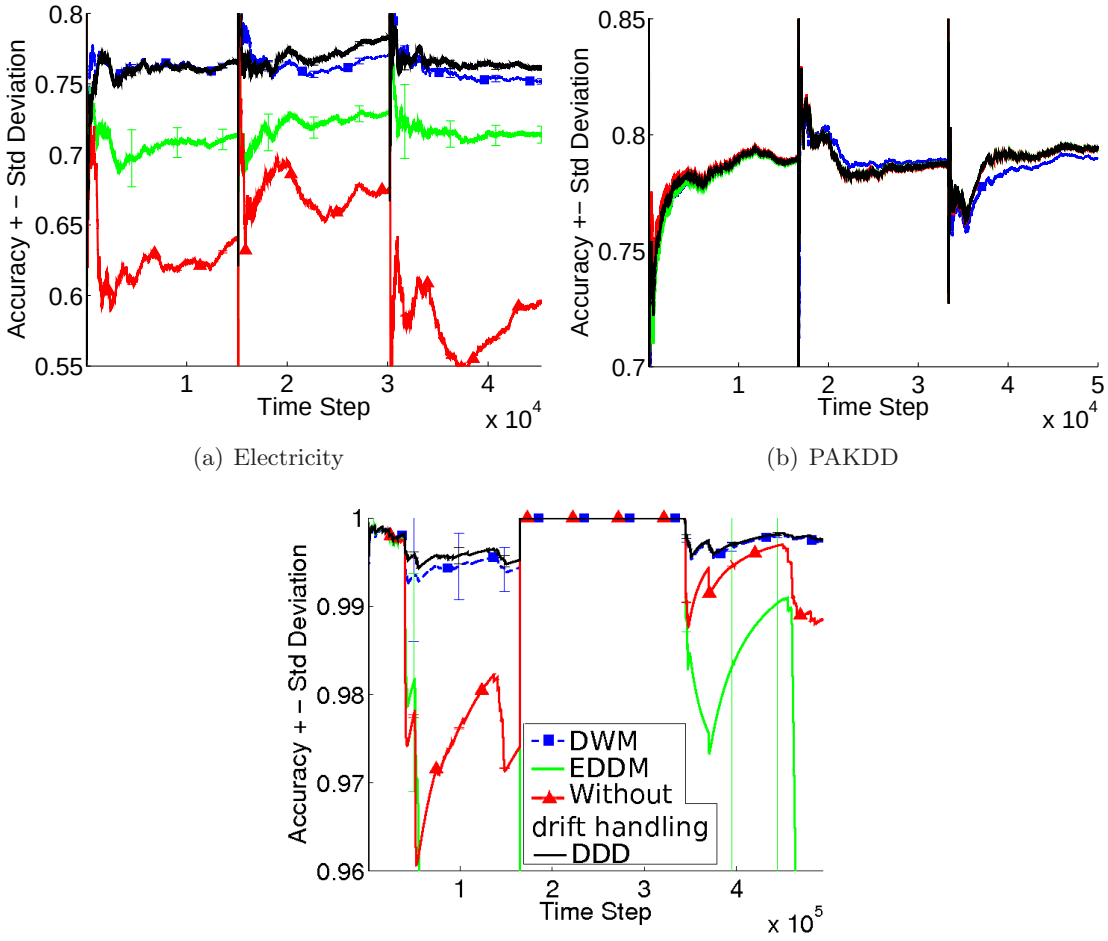
(c) KDD

Figure 6.10: Average prequential accuracy (equation 5.1) reset at every third of the learning, considering 30 runs using MLPs.

best false negative rates are different from the ones which obtain the best accuracy for all the approaches.

DDD can be easily adapted for dealing with minority classes by looking for inspiration from the skewed (imbalanced) data sets literature (Kotsiantis et al.; 2006). Increasing and decreasing diversity based on the parameter λ of the poisson distribution is directly related to sampling techniques. A $\lambda < 1$ can cause similar effect to under-sampling, whereas a $\lambda > 1$ can cause similar effect to over-sampling. Figure 6.12 shows the rate of false positives and negatives using $\lambda_l = \lambda_h = 2$ for the minority class, $\lambda_h = 0.005$ for the majority class, $\beta = 1.15$, $\alpha = 1.20$, $\rho = 0.3$, $p = 1$ both when using MLPs and NB, $\lambda_l = 0.4$ for the majority class when using MLPs and $\lambda_l = 0.1$ for the majority class when using NB. As we can see in figure 6.12, DDD obtained the best false negative rate.

The last data set analysed is KDD. In this problem, the probability of an intrusion considering the previous 2000 examples has several jumps from 1 to 0 and vice versa during the first and last third of the learning, as shown in figure 6.2(c). So, this data set probably presents several



(a) Electricity

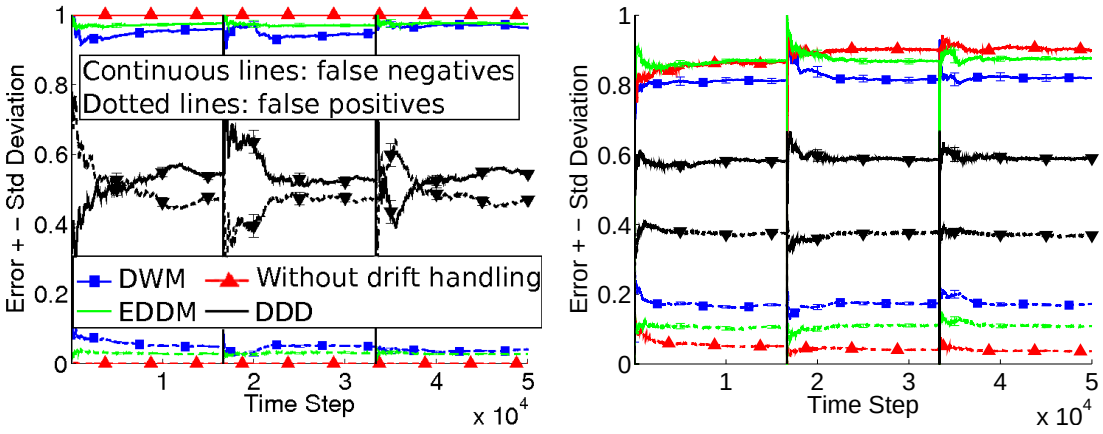
(b) PAKDD

(c) KDD

Figure 6.11: Average prequential accuracy (equation 5.1) reset at every third of the learning, considering 30 runs using NB.

severe and fast drifts which reoccur with a high rate during these periods. Even though DDD (and EDDM) is prepared for dealing with severe and fast drifts, it is not prepared for dealing with recurrent concepts yet. DWM does not have specific features to deal with recurrent drifts either, but it can obtain good accuracy in case these drifts are close enough to each other so that the weights of the base learners do not decay enough for them to be eliminated from the ensemble.

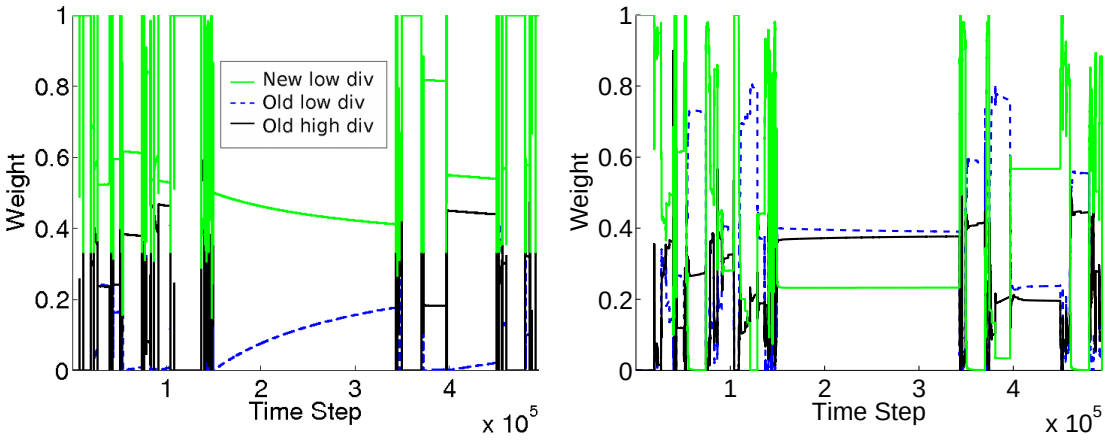
Figures 6.10(c) and 6.11(c) show that DDD obtained worse accuracy than DWM during the first and last thirds of the learning when using MLPs, but similar (or slightly better) when using NB. The analysis reveals that, if the drifts are very close to each other and there are no false alarms, DDD can make use of the learning of the old concept through the old ensembles (in particular the old low diversity) when the concept reoccurs. From figure 6.13(b), we can see that this is what happened when using NB, as the weight given to the old low diversity ensemble presents peaks during the learning and the number of drift detections (44) was consistent with the changes in the estimated prior probability of attack. However, false alarms can cause DDD to



(a) MLP (b) NB

Figure 6.12: Average false positive and negative error rates for PAKDD, reset at every third of the learning, considering 30 runs.

lose the ensembles which learnt the old concept (the old ensembles are replaced), being unable to use them when this concept reoccurs. This is what happened when using MLPs, as the number of drift detections (91) was more than the double of the number of detections when using NB, probably representing false alarms, and there were no peaks in the weight of the old low diversity ensemble (figure 6.13(a)) as when using NB.



(a) MLP (b) NB

Figure 6.13: Average weights used by DDD for KDD, considering 30 runs.

In summary, the experiments in this section reassure the analyses done in the previous sections: for a database which is likely to contain several continuous drifts (electricity), DDD attained better accuracy than EDDM and DWM. For a database which is likely to contain no drifts, DDD performed similarly to the other approaches. EDDM would perform worse than the other approaches if there were false alarms. For a database which may contain very severe and fast drifts reoccurring with a high rate, DDD performed similarly to DWM when it could make use of the ensembles which learnt the old concept, but worse when these ensembles were lost.

6.3.5 The Impact of the Parameters Choice on DDD's Accuracy

As commented in section 6.2, the parameter β of the drift detection method has greater influence on DDD's accuracy than the parameter λ, which controls the level of diversity used by DDD's high diversity ensembles. In this section, the impact of the parameters choice is further explained, based on the preliminary experiments with the real world data.

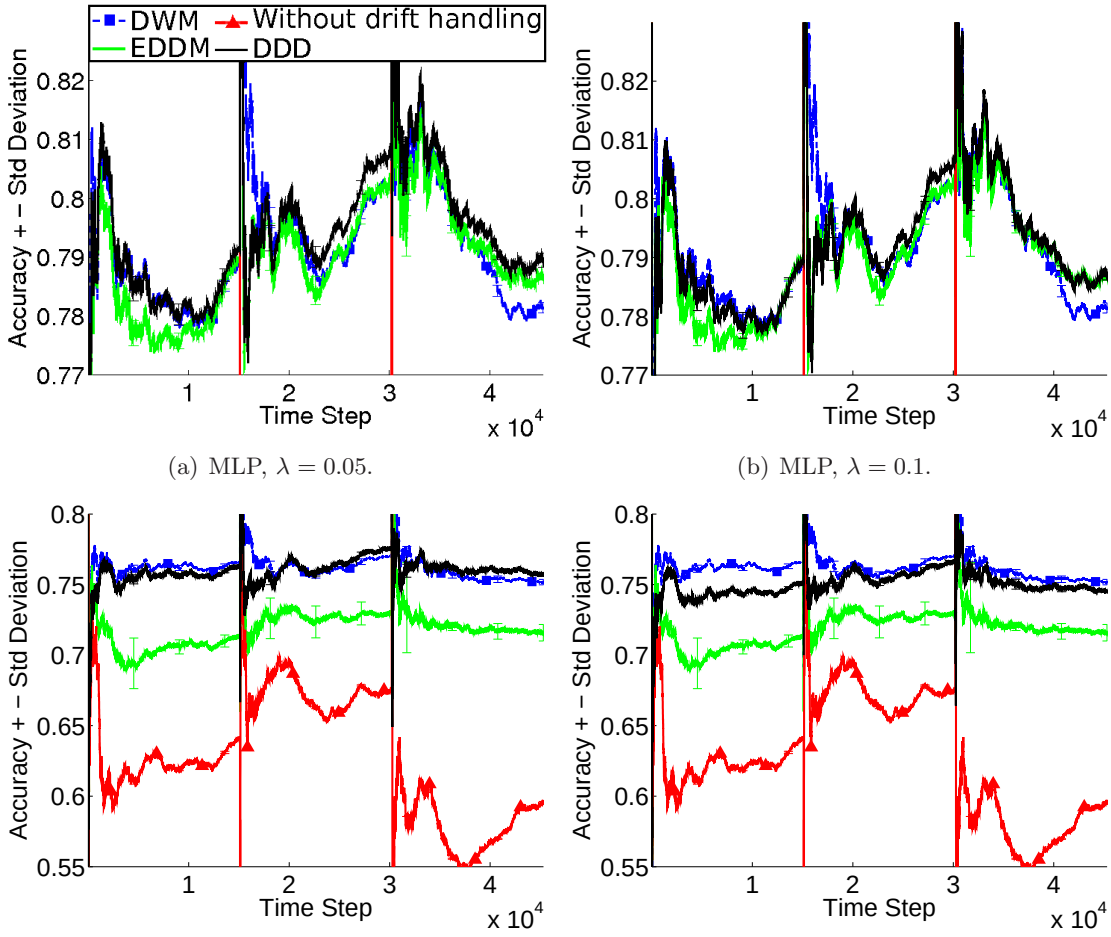
Firstly, the choice of β is commented on. Even though this parameter is not considered a DDD parameter, it has considerably high influence on DDD's accuracy. Low values such as 0.75 can cause drifts not to be detected and thus not treated by DDD, reducing its accuracy. Very high values such as 1.15 can cause so many false alarms when there are no drifts that W would have to be tuned not to reduce the accuracy. It is important to note, though, that even if W is kept with its default value of 1, DDD achieves considerably higher accuracy than EDDM when there are many false alarms. When it is not possible to tune β, this study recommends a value such as 0.95 to be used. The proposal of more precise drift detection methods is suggested as future work in the thesis.

Now, we shall analyse the influence of λ. This parameter controls the amount of diversity used by DDD's high diversity ensembles and can be considered a DDD parameter. According to the executions using the other best parameters from table 6.2(b), different values of $\lambda \in \{0.1, 0.05, 0.005\}$ never caused DDD's average accuracy to change from better to worse or vice versa in comparison to EDDM and DWM for PAKDD and KDD. The average accuracy changed slightly, but did not change the conclusions of the experiments.

For Electricity, different choices of λ affected the accuracy in such a way that DDD became similar or worse than the other approaches during some moments in which it was better when using the λs from table 6.2(b). It is important to emphasize that EDDM and DWM are using the best parameters among the preliminary executions in this comparison. As we can see in figure 6.14(a), when using MLPs, $\lambda = 0.05$ reduced DDD's accuracy slightly in comparison to figure 6.10(a), but the accuracy still remained higher than EDDM's and DWM's during most of the learning. When using $\lambda = 0.1$ (figure 6.14(b)), the accuracy dropped a bit further, so that DDD became more similar to the other approaches.

When using NB, $\lambda = 0.05$ (figure 6.14(c)) caused DDD's average accuracy to become slightly worse than DWM's during the first third of the learning. During the rest of the learning, even though DDD's and DWM's accuracies became more similar, DDD's accuracy was still higher during most of the period. When using $\lambda = 0.005$ (figure 6.14(d)), DDD's accuracy reduced further, becoming worse than DDD's, but still acceptable and better than EDDM's.

As we can see, different choices of λ can affect DDD's accuracy, but they are not likely to make it unacceptable. According to the experiments, the accuracy using different λs was usually not much lower than the average accuracies presented in figures 6.10 and 6.11. A more detailed study of the impact of the parameters choice on DDD's accuracy is proposed as future



(a) MLP, $\lambda = 0.05$.

(b) MLP, $\lambda = 0.1$.

(c) NB, $\lambda = 0.05$.

(d) NB, $\lambda = 0.005$.

Figure 6.14: Average accuracy for preliminary executions with the worst λ values according to the preliminary experiments. The other parameters are the same as in table 6.2(b).

work. The study should investigate not only the impact of each individual parameter, but also potential parameter interactions through the use of principled statistic methods, such as design of experiments and ANOVA.

6.4 Summary and Discussion

The main contribution of this chapter is a new approach to deal with concept drift using diversity. The approach is both accurate and robust in the presence of false positive drift detections in comparison to other approaches in the literature. It is an answer to research question 1.2.4: "is it possible to create an approach more robust and accurate considering different types of drift and, at the same time, achieve good accuracy in the absence of drifts?"

DDD maintains ensembles with different diversity levels, exploiting the advantages of diversity to handle drifts and using information from the old concept to aid the learning of the

new concept. None of the existing approaches in the literature fully uses the power of diversity for dealing with concept drift in such a way and neither use information from the old concept to aid the learning of the new concept. DDD is also designed to be more robust to false alarms than approaches which reset the system upon drift detection. Besides, it is designed to take immediate action to treat drifts once they are detected, instead of having to wait for weights which reflect the old concept to change and start reflecting the new concept, as approaches which do not use a drift detection method.

In the experiments done with several artificial and real world data sets, DDD attained better accuracy than EDDM mainly for drifts with low severity or low speed. The reason for the better accuracy is the use of the strategies explained in section 5.4. In that section, we verified that different strategies are better for different types of drift. In the present chapter, we verified that DDD is able to attribute considerably high weights for the most beneficial strategy. In this way, it achieved higher accuracy than EDDM, which always uses the same strategy of resetting the system upon drift detection.

When there were false alarms, DDD's accuracy was also better than EDDM's during stable concept, due to the use of old ensembles. DDD achieved considerably good robustness to false alarms using its default value for the parameter W and demonstrated to allow tuning the trade-off between robustness to false alarms and accuracy in the presence of drifts by changing W. EDDM's parameter α was not able to provide higher robustness to false alarms, as the number of consecutive training examples stored before the drift level was small even when very high α values were adopted. One of the possible ways to increase EDDM's accuracy on stable concepts would be to try to avoid false alarms by tuning the parameter β to a very low value. However, this would cause the approach to fail detecting drifts, reducing its accuracy in the presence of drifts. In this case, EDDM would not deal at all with the undetected drifts. The strategy of adopting a higher W for DDD in order to increase the robustness to false alarms is better, as it does not make it necessary to avoid false alarms under the risk of not detecting real drifts.

The analysis also revealed that false alarms can hinder the use of old ensembles in the case of sequences of recurrent drifts in which two concepts appear intercalated[1]. This is actually not a problem due to false alarms, but due to the fact that DDD, as most approaches in the literature, is not prepared to handle recurrent drifts. Had the recurrent concepts been more than two, the new drift detections would have eliminated old ensembles which could be used when a certain concept reappeared even if there were no false alarms. However, DDD can be adapted for recurrent drifts by moving old ensembles into a memory, instead of deleting them when detecting two consecutive drifts or when returning to the mode prior to drift detection. The use of memories is proposed as future work and includes investigations such as which ensembles

[1]It is worth remembering that a certain severity can be associated to the recurrence, as explained in chapter 4.1. In this case of two "intercalated" concepts, we consider that there can be a low severity in the recurrence, i.e., there could be returns to concepts very similar to previous concepts, instead of returns to exactly the same concepts.

should be included in the memory and how to limit the memory's size. The analysis from section 5.3.2.3 suggested that such a strategy is likely be successful.

Considering the experiments performed in this chapter, DDD was also almost always more accurate than DWM, both during stable concepts and after the drift, having usually faster recovery from drifts. DWM took longer time to recover from drifts possibly due to the maintenance of several base learners whose weights take time to start reflecting the new concept. Differently from DWM, the weights attributed by DDD to each of its ensembles are reset whenever a drift is detected, so that they can start reflecting the new concept more quickly. Moreover, DDD allows us to control the ensemble size. The fact that DWM automatically determines the ensemble size makes it vulnerable to failures in increasing the size enough to attain high accuracy. On the other extreme, as the size is not controlled, it could grow indefinitely and become too large.

From the above, we can conclude that, in all experiments carried out, DDD was overall accurate both in the presence and in the absence of drifts, with very few exceptions which occurred mainly in drifts with very high severity and speed and drifts with almost no change to the old concept. The main disadvantage of DDD is the need to maintain more ensembles. Nevertheless, even though DDD's time and memory requirements are higher, the time and memory complexity is the same as EDDM's.

DDD also demonstrated to be promising in the context of skewed distributions. This issue should be further investigated in order not to reduce the rate of false positives too much in detriment of the rate of false negatives.

Future work includes experiments using a parameter to control the maximum number of time steps maintaining four ensembles (aiming at reducing the amount of time maintaining four ensembles), further investigation of online environments with skewed distributions and extension of the approach to better deal with recurrent and predictable drifts. Another question to be dealt with is how to make better use of the high diversity ensemble before drift detection in DDD. As discussed in sections 5.3 and 5.4, this ensemble could be useful to reduce the initial drop in accuracy caused by a drift before the drift is detected. It was not used for predictions before drift detection by DDD in order to avoid reducing accuracy during stable concepts, when the low diversity ensemble is more accurate. Different ways to attribute weights for the ensembles could also be investigated, in order to improve accuracy soon after drift detections especially when drifts are very severe and sudden. The directions for the future work are explained in more detail in chapter 7.

Chapter 7

Conclusions and Future Work

This chapter summarizes the contributions of the thesis and gives directions for future work. The main contributions are answers to the research questions 1.2.1 to 1.2.4, in particular the study of ensemble learning in the presence of concept drift.

Chapter 3 presents a study of NCL in incremental and online learning, answering research question 1.2.1: "what are the strengths and weaknesses of negative correlation in incremental and online learning?" The chapter also illustrates one of the disadvantages of most incremental approaches existing in the literature: they are likely to be too plastic in detriment of the accuracy during stable concepts.

The study shows that NCL is promising in incremental environments. One of the proposed approaches, Growing NCL, is able to cope with forgetting. Both Growing NCL and Fixed Size NCL are able to get improvements in generalisation from the first to the last incremental step. However, reducing forgetting so as to achieve good overall generalisation is not straightforward. The experiments indicate that, even though Growing NCL is better at dealing with forgetting, it gets worse generalisation than Fixed Size NCL. A third approach, Selective NCL, combines features from the two previous approaches, managing to suffer less from forgetting than Fixed Size NCL and getting improved generalisation. Its main disadvantage is the higher training time, due to the GA-based selection procedure used to keep the ensemble with a fixed size.

Regarding online learning, chapter 3 shows that even though NCL has the advantage of having a parameter to tune diversity, it does not send a different sequence of training examples to each base learner. So, it has a restricted choice of base learners. This problem can be fixed by combining NCL with online bagging. Experiments show that the wider choice of base learners allows online bagging NCL to achieve better generalisation especially for the smaller databases.

The study also suggests not to use NCL for the diversity analysis presented in chapter 5, as the approach which sends a different sequence of training examples, online bagging NCL, has more than one source of diversity. In order to perform a principled study, it is desirable to use

an approach with a single source of diversity which can be tuned in such a way to consistently generate more or less diversity in the ensemble. So, a modified version of online bagging is analysed and used in chapter 5.

Chapter 5 provides answers to research questions 1.2.2 and 1.2.3: "when, how and why can ensembles be helpful for dealing with drifts?" and "can we use information from the old concept to better deal with the new concept? How?" The study concentrates on the impact of diversity on the performance of ensembles in online changing environments.

The analysis shows that, even though low diversity ensembles are likely to be more accurate on the old concept, high diversity ensembles can reduce the drop in accuracy on the new concept right after the beginning of the drift. The analysis also indicates that the higher the severity of the drift the most important the use of higher diversity if no additional strategy is adopted to converge to the new concept. This is particularly helpful when the system does not know that a drift is happening and cannot adopt any strategy to help convergence to the new concept.

Nevertheless, additional strategies are important to encourage convergence to the new concept. Different strategies in combination with diversity are analysed and the best strategy for each type of drift and moment of the learning is identified. In particular, the strategy of using an ensemble trained with high diversity on the old concept, but with low diversity on the new concept, is a way of using information from the old concept to aid the learning of the new concept. It is helpful to improve the accuracy when the drifts have low severity or speed.

Finally, chapter 6 uses these strategies to create a new approach for dealing with drifts, called Diversity for Dealing with Drifts (DDD). The approach is an answer to research question 1.2.4: "is it possible to create an approach more robust and accurate considering different types of drift and, at the same time, achieve good accuracy in the absence of drifts?" DDD manages to achieve good accuracy both in the presence and absence of drifts, thanks to its careful design to increase robustness to false alarms and to make better use of diversity for dealing with drifts. In the experiments with several artificial and real world data sets, it always obtained at least similar accuracy to EDDM and DWM both under several drift conditions and in the absence of drifts, with very few exceptions.

Future work includes a more detailed analysis of DDD's parameters, investigating not only the impact of each parameter by itself, but also possible parameter interactions using principled statistic methods, such as design of experiments and ANOVA. Investigation of better drift detection methods is also important and could use ensembles of drift detection methods in order to improve precision. The extension of DDD for dealing with recurrent and predictable drifts is also proposed as future work. For instance, the use of memories to store old ensembles that can be retrieved when useful should be investigated. DDD's behaviour for problems with skewed distributions should also be further analysed and improved. A very different research direction could also be taken, by trying to detect the type of drift and then adopting the most appropriate strategy for that type of drift.

The next sections explain the contributions given by each chapter from 3 to 6 and the future work in more detail.

7.1 Negative Correlation in Incremental and Online Learning

7.1.1 Contributions

NCL has shown to be able to outperform other ensemble methods in offline mode, suggesting that it might also be helpful in incremental or online learning. However, there has been no study of NCL in these learning modes before the thesis. So, the thesis deals with the following research question: "what are the strengths and weaknesses of negative correlation in incremental and online learning?" (research question 1.2.1). It shows that NCL is promising for incremental learning. Good results can also be obtained in online learning, by combining NCL to other approaches such as online bagging.

Section 3.1 shows that two possible ways to use NCL in incremental learning are Growing NCL and Fixed Size NCL. Growing NCL does not use old learners to learn new data, being able to overcome forgetting. However, it gets lower generalisation than Fixed Size NCL. The reason for the lower generalisation is likely to be because it uses only one base learner to learn each new chunk of data. Fixed Size NCL uses several base learners and allows all of them to learn new data. It is able to achieve higher generalisation, but the experiments indicate that it suffers more from forgetting.

A relationship between low/high forgetting and high/low improvement in generalisation was observed in Fixed Size NCL, suggesting that reducing forgetting may further improve its generalisation. This relationship was not observed in Growing NCL because its generalisation is hindered by the use of only one neural network to learn new data. So, reducing forgetting so as to achieve better generalisation is not a straightforward task.

A third approach proposed in a work in collaboration, Selective NCL, combines some features of Growing NCL to Fixed Size NCL. It was able to slightly improve generalisation in comparison to the two former approaches, at the same time as it suffered less from forgetting than Fixed Size NCL. Another advantage of Selective NCL over Growing NCL and Learn++ is its fixed ensemble size. Its main disadvantage is the training time, which becomes higher because of the GA-based selection process.

The study of incremental NCL also illustrates a problem presented by most existing incremental learning approaches designed for changing environments. When allowing only a new base learner to learn a new chunk of data, they give high emphasis in plasticity, which is good for frequent drifts. However, when the drifts are less frequent than the chunk size, these approaches are likely to get lower accuracy than if each training example contributed to the learning of sev-

eral ensemble members. Well designed strategies to deal with drifts allowing several ensemble members to learn each new chunk of data are thus more desirable.

In the context of online learning, section 3.2 shows that NCL is less helpful when applied by itself, in particular if the databases are not large, because of the restricted choice of base learners. However, when combined with online bagging, it sends a different sequence of data to each base learner, allowing a wider range of choices for the base learner. In this way, it is able to achieve good accuracy through the use of more appropriate base learners, especially for smaller data sets.

7.1.2 Future Work

Selective NCL's accuracy should be further improved to outperform other approaches such as Learn++ and to reduce the training time. This may be achieved by adopting another selection procedure for the base learners. In particular, inspiration can be taken from the feature selection literature (Guyon and Elisseeff; 2003; Molina et al.; 2002) by making analogies between features and ensemble members.

Another point to be studied is the influence of the penalty term in online learning in comparison to offline learning. We saw that online MLPs are likely to get worse generalisation than offline MLPs particularly for smaller databases because they use only one epoch. As the number of epochs also influences how many times the NCL penalty term is used, it is important to verify if the help provided by NCL to increase the accuracy is also hindered.

Some additional preliminary experiments not included in chapter 3 suggest that the penalty term may indeed be less helpful in online learning than in offline learning. The experiments used the direct application of NCL and 10 databases from the UCI Machine Learning Repository, including the ones used in section 3.2. The MLPs used two different setups: 6 and 10 hidden nodes. Ensemble sizes of 5 and 10 MLPs, with penalty strength γ of 0 and 0.5 were also used. The learning rates were 0.1 for all databases but Mushroom, in which it was 0.05. The number of epochs for the offline MLPs varied among 10, 50, 100, 300 and 1500, depending on which number gave the best results. For the online MLPs, a single epoch was used. Five runs of 2-fold cross-validation were performed in order to use 5x2 cross-validation F tests with 95% of confidence (Dietterich; 1998; Alpaydin; 1999) for the comparisons.

The experiments revealed that, when using offline learning, $\gamma = 0.5$ helped to improve the accuracy in 6 out of 10 databases in comparison to $\gamma = 0$. In online learning, this number dropped to 4 in 10. Besides, when using offline learning, the improvement usually occurred for several parameter choices for each database. For online learning, the improvement usually occurred for only one of the parameter choices, suggesting that it is more difficult to tune the parameters in online mode.

This issue should be further investigated using additional parameter values and checking whether higher penalty strengths in online mode could compensate the fact that the penalty term is used less times. The influence of the penalty term in online bagging NCL should also be better investigated.

7.2 Preparing Principled Studies of Concept Drift

7.2.1 Contributions

When studying drifts, it is important to determine how existing and new approaches or strategies behave considering different types of drift. So, in order to perform principled studies, it is necessary to use a clear categorisation, separating drifts according to different criteria into mutually exclusive and non-heterogeneous categories. However, the literature lacks a clear and systematic categorisation. Very few and highly heterogeneous categories are used, separating drifts according to only two criteria (speed and recurrence) and ignoring other important features.

Chapter 4 provides a new categorisation inspired by the dynamic optimisation problems literature (Branke; 1999, 2002; Yaochu and Branke; 2005). It considers the criteria severity, speed, predictability, frequency and recurrence. The new criteria, together with the revocation of the term "intermediate concept", allow consistent characterization of drifts, creating mutually exclusive and non-vague categories. The importance of the categorisation is further demonstrated in chapters 5 and 6 when strategies behave differently for drifts that would not be differentiated through the categorisation existing in the literature.

When evaluating strategies and approaches for dealing with drifts, it is also important to use artificial data sets in which we know which types of drift are present and when they occur. In this way, we can determine to which types of drift certain strategies or approaches are better or worse, revealing when they can be helpful and which points need to be improved. After using artificial data sets, it is still important to use real world data sets to further confirm the usefulness of the approaches.

The benchmarks used in the literature, such as SEA (Street and Kim; 2001) and STAGGER (Schlimmer and Granger Jr.; 1986) concepts, do not contain enough variety of drifts to allow principled and detailed studies. Chapter 4 provides new data sets based on existing benchmarks. The data sets simulate drifts with low, medium and high severity and speed, allowing principled studies of drift.

7.2.2 Future Work

The data sets presented in chapter 4 contain isolated drifts with different levels of severity and speed. For studies which do not emphasize predictable and recurrent drifts, these data sets are enough, as sequences of drift can be analysed through real world problems. However, studies of recurrent and predictable drifts would benefit from data sets containing different sequences of different types of recurrent and predictable drifts. So, such data sets should be created as future work.

7.3 A Diversity Study in the Presence of Drift

7.3.1 Contributions

Even though ensembles have been used to handle concept drift, the literature did not contain any deep study of why they can be helpful for that and which of their features can contribute or not to deal with concept drift. Such a study is important because a better understanding of the behaviour of ensembles in the presence of concept drift allows better exploitation of their features for dealing with drifts. Chapter 5 presents such a study, answering research question 1.2.2: "when, how and why ensembles can be helpful for dealing with drifts?"

The study shows that, even though low diversity ensembles are likely to be more accurate on the old concept, high diversity ensembles can reduce the drop in accuracy on the new concept right after the beginning of the drift. The analysis also indicates that the higher the severity of the drift the more important the use of higher diversity if no additional strategy is adopted to converge to the new concept. This is particularly helpful when the system does not know that a drift is happening and cannot adopt any strategy to help convergence to the new concept.

Nevertheless, additional strategies are important to encourage convergence to the new concept. Simply training on the new concept an old ensemble trained with low diversity on the old concept is helpful during part of the drifting time in gradual drifts, as some of the instances to be predicted still reflect the old concept. The strategy of using an ensemble trained with high diversity on the old concept, but with low diversity on the new concept, is helpful to improve the accuracy when the drifts have low severity or speed. This behaviour is intuitively reasonable, as the old concept can be helpful when the changes are slow or not large. When the drift is fast and with high severity, the best strategy is to simply create a new low diversity ensemble from scratch to learn the new concept.

This study also allows us to answer research question 1.2.3: "can we use information from the old concept to better deal with the new concept? How?" Intuitively, if a concept drift causes few changes to the old concept, information learnt from the old concept should be helpful to aid the learning of the new concept. However, to the best of our knowledge, no approach in the

literature attempts to use information from the old concept in order to *aid* the learning of the new concept. Chapter 5 shows that using an ensemble trained with high diversity on the old concept, but with low diversity on the new concept, allows the use of information from the old concept to aid the learning of the new concept.

An additional contribution given by chapter 5 is the modified online bagging algorithm. The study indicates that the simple technique of using a pre-defined parameter for the *Poisson* distribution of online bagging allows us to consistently and directly encourage more or less diversity in the ensemble.

7.3.2 Future Work

Further study of diversity in the presence of recurrent and predictable drifts could be done as future work.

It is also interesting to check whether the difficulty of the concept learnt before a drift influences the learning of the new concept. As explained in section 5.3, the study raised suspicions that a concept easily learnt may be more difficult to be forgotten, possibly increasing the error on the new concept. Another hypothesis to be checked is if higher diversity can help learning when there are many irrelevant attributes, as the only database in which a λ lower than 1 ($\lambda = 0.1$) provided the best test error before the drift was Plane, which contains many irrelevant attributes.

A different direction of research related to this study is to determine whether it is possible to estimate the underlying distributions of a certain concept so as to identify it when it reappears. In this way, information learnt from a certain concept could be reused if and when the concept reappears. The use of naive Bayes as base learners may be helpful for that, as it is a direct way to estimate the posterior probability.

Further directions also include the study of other features which may help to deal with drifts, such as ensemble size, base learners' individual accuracy and feature selection. Considering feature selection, as an initial study, the effect of ensembles with different sets of features could be analysed. That could have a relationship with diversity, which should be verified.

Another direction would be to investigate the use of mutation to deal with concept drifts. Very high diversity does not allow good convergence to the current concept, allowing the ensemble to converge to a new concept at the same time as information from the old concept can be used to aid the learning. Instead of using an ensemble with high diversity, we could mutate the learners. An example of mutation for MLPs would be to add small random values to its weights. The disadvantage would be that different types of mutation would be required for different types of learners.

7.4 Diversity for Dealing with Drifts

7.4.1 Contributions

Chapter 5 shows how diversity and other strategies can be used to help dealing with each type of concept drift. However, in a real world situation, we do not know what sort of drift is present in the data stream. So, we are still left with the research question presented in section 1.2.4: "is it possible to create an approach more robust and accurate considering different types of drift and, at the same time, achieve good accuracy in the absence of drifts?"

This question is answered in chapter 6, through the proposal of a new approach called Diversity for Dealing with Drifts (DDD). The approach maintains ensembles with different diversity levels, exploiting the advantages of diversity to handle drifts and using information from the old concept to aid the learning of the new concept. None of the existing approaches in the literature fully uses the power of diversity for dealing with concept drift in such a way and neither use information from the old concept to aid the learning of the new concept. DDD is also designed to be more robust against false alarms than approaches which reset the system upon drift detection. Besides, immediate action to treat drifts is taken once they are detected, instead of having to wait for weights which reflect the old concept to change and start reflecting the new concept, as approaches which do not use a drift detection method.

In the experiments with several artificial and real world data sets, DDD obtained at least similar accuracy to EDDM and DWM both under several drift conditions and in the absence of drifts, with very few exceptions.

In comparison to EDDM, the exceptions occurred in a very few of the drifts with high severity and speed, due to the initial weights attributed to the ensembles. In comparison to DWM, the exceptions occurred mainly when the drifts were detected, but had almost no influence on the accuracy. Higher accuracy than EDDM was obtained mainly for drifts with low severity or low speed, thanks to the use of the strategies explained in section 5.4. DDD is also more robust against false alarms than EDDM, achieving higher accuracy when they occur, in particular during stable concepts. DDD's accuracy was almost always higher than DWM's.

Another advantage of DDD over DWM is its fixed ensemble size. The ensembles do not grow indefinitely as more data becomes available and their size can be large enough not to have the accuracy affected badly. Even though DDD maintains more ensembles than EDDM, the time and memory complexity class of these two approaches is the same.

DDD also demonstrated to be promising in the context of skewed distributions.

7.4.2 Future Work

DDD, as well as many other drift handling approaches in the literature, does not have a strategy for dealing with recurrent and predictable drifts. However, it can be adapted for recurrent drifts by moving old ensembles into a memory, instead of deleting them when detecting two consecutive drifts or when returning to the mode prior to drift detection. This work would include investigations such as which ensembles to include in the memory and how to limit the size of the memory. Ensembles stored in memory could be retrieved based on their accuracy on the current concept, or based on estimations of the underlying distributions, as suggested in section 7.3.2.

The analysis from section 5.3.2.3 suggests that memories are likely be successful, as ensembles with lower diversity managed to get improved accuracy when partial returns to previous concepts occurred. Memories may also be useful for predictable drifts. An additional learning machine could be used to predict drifts and to determine which base learners in memory could be helpful. In the dynamic optimisation problems literature, estimators such as Kalman Filter have been used to predict changes (Rossi et al.; 2007).

Further investigation of online environments with skewed distributions should also be done. In particular, it may be important to detect drifts separately in the context of the minority and majority classes. The reason for that is that changes in the minority classes may not be detected if we consider some measure related to the overall accuracy. Strategies not to reduce the rate of false positives too much in detriment of the rate of false negatives should also be investigated.

Other issues to be studied are related to the improvement of DDD's overall behaviour. For example, the possibility of reducing the time maintaining four ensembles could be analysed. Experiments could be done using a parameter to control the maximum number of time steps maintaining four ensembles. The strategy of mutating base learners, commented on section 7.3.2, could also be viewed as way to reduce the number of ensembles. If such a strategy is successful, it would be possible to maintain a single ensemble or even a single learner, if desired, instead of maintaining two or four ensembles.

Besides, DDD does not use the high diversity ensemble for predictions before drift detection. This design choice was made in order to avoid reducing the accuracy during stable concepts, as the low diversity ensemble is more accurate in this case. Nevertheless, as discussed in sections 5.3 and 5.4, the high diversity ensemble could be useful to reduce the initial drop in accuracy caused by a drift before the drift is detected. If this ensemble is maintained in DDD, it would be good to check how to use it to help increase the accuracy not only after, but also before drift detections. This work could be linked with the work about drifts prediction. For example, if a drift is predicted, the high diversity ensemble could be activated before the drift is confirmed.

Different ways to attribute weights for the ensembles could also be investigated, in order to improve accuracy soon after drift detection especially when drifts are very severe and sudden.

The experiments showed that, even though in a very few cases, the inaccuracy of the weights soon after drift detection due to the small number of training examples used to calculate them caused DDD to get lower accuracy than EDDM. The difficulty of estimating the accuracy of rules when the number of available performance samples is small is also a known problem in the probability smoothing and Bayesian approaches literature (Marshall et al.; 2007). So, this area might be related and could be studied to check whether it can inspire other weighting schemes for DDD.

DDD's accuracy is also influenced by the drift detection method. A method which can detect the drifts early and accurately would improve DDD's accuracy. So, the proposal of better drift detection methods should also be investigated. For instance, the idea of using an ensemble of drift detection methods existing in the literature could be analysed. As ensembles of learning machines are helpful for improving generalisation, they may also be helpful to provide more accurate drift detections. Another idea would be to analyse the estimate of the underlying distributions directly in order to detect drifts, instead of analysing some measure related to accuracy.

A more detailed study of the impact of the parameters choice on DDD's accuracy is also proposed as future work. The study should investigate not only the impact of each individual parameter, but also potential parameter interactions through the use of principled statistic methods, such as design of experiments and ANOVA.

The investigation of domain knowledge incorporation and how much it can increase DDD's accuracy for specific domains could also be investigated.

Finally, as commented in the beginning of chapter 6, in order to answer research question 1.2.4, we could (1) design a drift detection method able to identify the type of drift and then use the most appropriate strategy according to section 5.4 or (2) propose an approach which is more robust for several types of drift without necessarily identifying the type of drift. The thesis concentrates on the latter, but the investigation of the former could be done as future work.

Appendix A

Evolving Fuzzy Neural Networks (EFuNNs)

EFuNNs (Kasabov; 2001) are online neural networks. They work with one pass through the training examples and perform local and constructive learning. Local and constructive learning are considered important in an attempt to avoid catastrophic forgetting.

EFuNNs have a five-layer architecture. The first layer represents the input vector, the second represents the fuzzy quantification of the input vector, the third represents the associations between fuzzy input space and fuzzy output space, the fourth represents the fuzzy quantification of the output vector and the fifth represents the output vector.

Learning occurs at the rule nodes layer. Each node *rj* of this layer is represented by two vectors of connection weights (*W1(rj)* and *W2(rj)*). *W1* represents the coordinates of the nodes in the fuzzy input space and it is adjusted through unsupervised learning. *W2* represents the coordinates of the nodes in the fuzzy output space and it is adjusted through supervised learning. The learning rules are the following:

- $W1(r_j) = W1(r_j) + lr1(r_j) * (x_f(d) - W1(r_j))$
- $W2(r_j) = W2(r_j) + lr2(r_j) * (t_f(d) - A2) * A1(r_j)$

where: $x_f(d)$ and $t_f(d)$ are the fuzzy input and fuzzy output vectors of the training pattern d; $lr1(r_j)$ and $lr2(r_j)$ are the learning rates for the $W1$ and $W2$ weights of the node r_j at a particular time during the learning; $A2$ is the fuzzy output activation vector and $A1(r_j)$ is the activation value of the rule node r_j. The learning rate of a node can be the inverse of the number of training patterns accommodated so far by that node.

Algorithm A.6 briefly describes EFuNN' learning algorithm. For more details, it is recommended to read (Kasabov; 2001).

Algorithm A.6 EFuNN

Inputs: current EFuNN, training pattern d, number of training patterns presented so far and training parameters (number of membership functions; type of membership functions; initial sensitivity threshold S of the nodes, which is also used to determine the initial radius of the receptive field of a node r_j when it is created ($R(r_j) = 1 - S$); error threshold E; aggregation parameter $Nagg$; pruning parameters OLD and Pr; *m-of-n* value, which is the number of highest activation nodes used in the learning; maximum radius of the receptive field $Mrad$; rule extraction thresholds $T1$ and $T2$).

1: **if** this is the first learning of EFuNN **then**
2: Set the first rule node r_0 to memorize d: $W1(r_0) = x_f(d)$ and $W2(r_0) = t_f(d)$.
3: **else**
4: Calculate the activations $A1$ of all rule nodes, e.g., $A1 = 1 - D(W1(r_j), x_f(d))$, where D is a distance measure.
5: Select the rule node r_k that has the smallest distance $D(W1(r_k), x_f(d))$ and that has activation $A1(r_k) >= S(r_k)$. In the case of *m-of-n* learning, select m nodes instead of just one node.
6: **if** such a node does not exist **then**
7: Create a new rule node.
8: **else**
9: Determine the activation $A2$ of the output layer and the normalized output error $Err = subabs(t(d), F_{ef})/Nout$, where $t(d)$ is the desired output, F_{ef} is the obtained output and $Nout$ is the number of nodes of the output layer.
10: **if** $Err > E$ **then**
11: Create a new rule node.
12: **else**
13: Apply the learning rules to $W1(r_k)$ and $W2(r_k)$ (in the case of *m-of-n* learning, the rules are applied to the m rule nodes).
14: **end if**
15: Apply aggregation procedure after the presentation of $Nagg$ patterns.
16: Update the parameters $S(r_k)$, $R(r_k)$, $Age(r_k)$ and $TA(r_k)$. $TA(r_k)$ can be, for example, the sum of the activations $A1$ obtained for all examples that r_k accommodates.
17: Prune rule nodes according to OLD and Pr.
18: Extract rules according to $T1$ and $T2$.
19: **end if**
20: **end if**

List of References

Abdulsalam, H., Skillicorn, D. B. and Martin, P. (2007). Streaming random forests, *Proc. of IDEAS*, Banff, Canada, pp. 225–232.

Adamczak, R., Duch, W. and Jankowski, N. (1997). New developments in the feature space mapping model, *Proceedings of the Third Conference on Neural Networks and Their Applications*, Kule, Poland, pp. 65–70.

Aha, D., Kibler, D. and Albert, M. K. (1991). Instance-based learning algorithms, **6**: 37–66.

Ali, H. A., El Desouky, A. I. and Saleh, A. I. (2006). *Agent Technologies and Web Engineering: Applications and Systems*, Information Science Publishing, chapter Studying and Analysis of a Vertical Web Page Classifier Based on Continuous Learning Nave Bayes (CLNB) Algorithm, pp. 210–254.

Alpaydin, E. (1999). Combined 5x2cv f test for comparing supervised classification learning algorithms, *Neural Computation* **11**: 1885–1892.

Angelov, P. (2006). Nature-inspired methods for knowledge generation from data in real-time. **URL:** *http://www.nisis.risk-technologies.com/popup/Mallorca2006_Papers/A333_13774_Nature-inspiredmethodsforKnowledgeGeneration_Angelov.pdf*

Araujo, L. and Merelo, J. J. (2007). A genetic algorithm for dynamic modelling and prediction of activity in document streams, *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, ACM, London, pp. 1896–1903.

Baena-García, M., Del Campo-Ávila, J., Fidalgo, R. and Bifet, A. (2006). Early drift detection method, *Proceedings of the Forth ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS'06)*, Berlin, Germany, pp. 77–86.

Baguley, T. (2004). An introduction to sphericity. Nottingham Trent University. **URL:** *http://homepages.gold.ac.uk/aphome/spheric.html*

Berry, D. A. and Fristedt, B. (1985). *Bandit Problems: Sequential Allocation of Experiments (Monographs on Statistics and Applied Probability)*, Chapman & Hall.

Bishop, C. M. (2005). *Neural Networks for Pattern Recognition*, Oxford University Press, United Kingdom.

Blum, A. (1997). Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain, *Machine Learning* **26**: 5–23.

Branke, J. (1999). Evolutionary algorithms for dynamic optimization problems - a survey, *Technical Report 387*, Insitute AIFB, University of Karlsruhe.

Branke, J. (2002). *Evolutionary Optimization in Dynamic Environments*, Kluwer Academic Publishers, Netherlands.

Branke, J., Salihoglu, E. and Uyar, S. (2005). Towards an analysis of dynamic environments, *Proceedings of the Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*, ACM, USA, pp. 1433–1440.

Breiman, L. (1996a). Bagging predictors, *Machine Learning* **24**(2): 123–140.

Breiman, L. (1996b). Bias, variance, and arcing classifiers, *Technical Report 460*, Department of Statistics, University of California, Berkeley, USA.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line, *Machine Learning* **36**: 86–103.

Breiman, L. (2001). Random forests, *Machine Learning* **45**: 5–32.

Brown, G. (2004). *Diversity in Neural Network Ensembles*, PhD thesis, School of Computer Science, The University of Birmingham, Birmingham, UK.
URL: *http://www.cs.man.ac.uk/∼gbrown/research.php*

Brown, G., Wyatt, J., Harris, R. and Yao, X. (2005). Diversity creation methods: A survey and categorisation, *Journal of Information Fusion* **6**: 5–20.

Brown, G., Wyatt, J. L. and Tiňo, P. (2005). Managing diversity in regression ensembles, *Journal of Machine Learning Research* **6**: 1621–1650.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H. and Rosen, B. (1992). Fuzzy ARTMAP: A neural network for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on Neural Networks* **3**: 698–713.

Chandra, A., Chen, H. and Yao, X. (2006). *Multi-objective Machine Learning*, Springer-Verlag, chapter Trade-off Between Diversity and Accuracy in Ensemble Generation, pp. 429–464.

Chandra, A. and Yao, X. (2006). Evolving hybrid ensembles of learning machines for better generalisation, *Neurocomputing* **69**: 686–700.

Chen, H. and Yao, X. (2009). Regularized negative correlation learning for neural network ensembles, *IEEE Transactions on Neural Networks* **20**(12): 1962–1979.

Chu, F. and Zaniolo, C. (2004). Fast and light boosting for adaptive mining of data streams, *Proceedings of the Eight Pacific-Asia Knowledge Discovery and Data Mining Conference (PAKDD'04)*, Sydney, pp. 282–292.

Dawid, A. and Vovk, V. (1999). Prequential probability: Principles and properties, *Bernoulli* **5**(1): 125–162.

De Jong, K. (2006). *Evolutionary Computation: A unified approach*, MIT Press, Cambridge.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* **7**: 1–30.

Dietterich, T. G. (1997). Machine learning research: Four current directions, *Artificial Intelligence* **18**(4): 97–136.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms, *Neural Computation* **10**: 1895–1923.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, *Machine Learning* **40**(2): 139–157.

Domingo, C. and Watanabe, O. (2000). Madaboost: A modification of adaboost, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, Morgan Kaufmann Publishers Inc, San Francisco, pp. 180–189.

Drucker, H., Schapire, R. E. and Simard, P. (1992). Improving performance in neural networks using a boosting algorithm, *Proceedings of the Fifth Advances in Neural Information Processing Systems Conference (NIPS'92)*, Morgan Kaufmann, San Francisco, pp. 42–49.

Fan, W. (2004). Streamminer: a classifier ensemble-based engine to mine concept-drifting data streams, *Proceedings of the 30th International Conference on Very Large Data Bases*, Toronto, pp. 1257–1260.

Fern, A. and Givan, R. (2000). Online ensemble learning: An empirical study, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, Morgan Kaufmann, San Francisco, pp. 279–286.

Fern, A. and Givan, R. (2003). Online ensemble learning: An empirical study, *Machine Learning* **53**: 71–109.

Folino, G., Pizzuti, C. and Spezzano, G. (2007). An adaptive distributed ensemble approach to mine concept-drifting data streams, *Proceedings of the Nineteenth IEEE International Conference on Tools with Artificial Intelligence*, Vol. 2, Patras, Greece, pp. 183–188.

Forman, G. (2006). Tackling concept drift by temporal inductive transfer, *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, pp. 252–259.

Freund, Y. (1995). Boosting a weak learning algorithm by majority, *Information and Computation* **121**: 256–285.

Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55**(1): 119–139.

Freund, Y. and Schapire, R. E. (1996a). Experiments with a new boosting algorithm, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, Morgan Kaufmann, Bari, Italy, pp. 148–156.

Freund, Y. and Schapire, R. E. (1996b). Game theory, on-line prediction and boosting, *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, ACM Press, New York, pp. 325–332.

Gama, J. and Kosina, P. (2009). Tracking recurring concepts with meta-learners, *Proceedings of the Fourteenth Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*, Aveiro, Portugal, pp. 423–434.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P. (2004). Learning with drift detection, *Proceedings of the Seventh Brazilian Symposium on Artificial Intelligence (SBIA'04) - Lecture Notes in Computer Science*, Vol. 3171, Springer, São Luiz do Maranhão, Brazil, pp. 286–295.

Gao, J., Fan, W. and Han, J. (2007). On appropriate assumptions to mine data streams: Analysis and practice, *Proceedings of the IEEE International Conference on Data Mining (ICDM'07)*, Omaha, NE, pp. 143–152.

Gao, J., Fan, W., Han, J. and Yu, P. (2007). A general framework for mining concept-drifting data streams with skewed distributions, *Proceedings of the 2007 SIAM International Conference on Data Mining*, Minneapolis, Minnesota.

Greenhouse, S. and Geisser, S. (1954). On methods in the analysis of profile data, *Psychometrika* **24**: 95–112.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection, *Journal of Machine Learning Research* **3**: 1157–1182.

Harries, M. (1999). Splice-2 comparative evaluation: Electricity pricing, *Technical Report UNSW-CSE-TR-9905*, Artificial Intelligence Group, School of Computer Science and Engineering, The University of New South Wales, Sidney.

He, H. and Chen, S. (2008). Imorl: Incremental multiple-object recognition and localization, *IEEE Transactions on Neural Networks* **19**(10): 1727–1738.

Hettich, S. and Bay, S. D. (1999). The UCI KDD archive. University of California, Irvine, Dept. of Information and Computer Sciences.
URL: *http://kdd.ics.uci.edu*

Howell, D. (2007). *Statistical Methods for Psychology*, Thomson Wadsworth, Belmont, California.

Inoue, H. and Narihisa, H. (2000). Improving generalization ability of self-generating neural networks through ensemble averaging, *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining (LNAI 1805)*, Kyoto, Japan, pp. 177–180.

Inoue, H. and Narihisa, H. (2003). Effective pruning method for a multiple classifier system based on self-generating neural networks, *Proceedings of the 2003 Joint International Conference (ICANN/ICONIP'03 - LNCS 2714)*, Istanbul, Turkey, pp. 11–18.

Inoue, H. and Narihisa, H. (2005). Self-organizing neural grove and its applications, *Proceedings of the 2005 International Joint Conference on Neural Networks (IJCNN'05)*, Vol. 2, Montreal, Canada, pp. 1205–1210.

Islam, M., Yao, X. and Murase, K. (2003). A constructive algorithm for training cooperative neural network ensembles, *IEEE Transactions on Neural Networks* **14**(4): 820–834.

Kasabov, N. (2001). Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics* **31**(6): 902–918.

Kasabov, N. (2003). *Evolving Connectionist Systems*, Springer, Great Britain.

Kasabov, N. and Song, Q. (2002). Denfis: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction, *IEEE Transactions on Fuzzy Systems* **10**(2): 144–154.

Kelly, M. G., Hand, D. J. and Adams, N. M. (1999). The impact of changing populations on classifier performance, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD'99)*, ACM, New York, pp. 367–371.

Kohonen, T. (1995). *Self-Organizing Maps*, Springer-Verlag, Berlin.

Kolter, J. Z. and Maloof, M. A. (2003). Dynamic weighted majority: A new ensemble method for tracking concept drift, *Proceedings of the Third International IEEE Conference on Data Mining (ICDM'03)*, IEEE Press, Los Alamitos, CA, pp. 123–130.

Kolter, J. Z. and Maloof, M. A. (2005). Using additive expert ensembles to cope with concept drift, *Proceedings of the Twenty Second ACM International Conference on Machine Learning (ICML'05)*, Bonn, Germany, pp. 449–456.

Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts, *Journal of Machine Learning Research* **8**: 2755–2790.

Kotsiantis, S. B. and Pintelas, P. E. (2004). An online ensemble of classifiers, *Proceedings of the Fourth International Workshop on Pattern Recognition in Information Systems (PRIS'04)*, INSTICC Press, Porto, Portugal, pp. 59–68.

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006). Handling imbalanced datasets: A review, *GESTS International Transactions on Computer Science and Engineering* **30**(1): 25–36.

Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning, *Proceedings of the Eighth Advances in Neural Information Processing Systems Conference (NIPS'95)*, Vol. 7, pp. 231–238.

Kuh, A., Petsche, T. and Rivest, R. L. (1990). Learning time-varying concepts, *Proceedings of the Third Conference on Advances in Neural Information Processing Systems (NIPS'90)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, pp. 183–189.

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, *Machine Learning* **51**: 181–207.

Lane, D., Lu, J., Peres, C. and Zitek, E. (2008). Online statistics: An interactive multimedia course of study.
URL: *http://onlinestatbook.com/index.html*

Larose, D. T. (2004). *Discovering Knowledge in Data: An Introduction to Data Mining*, Wiley-Interscience.

LeCun, Y., Bottou, L., Orr, G. B. and Muller, K.-R. (1998). Efficient BackProp, *Neural networks: tricks of the trade*, Springer, Berlin, p. 44p.

Lee, H. K. H. and Clyde, M. A. (2004). Lossless online bayesian bagging, *Journal of Machine Learning Research* **5**: 143–151.

Lim, C. P. and Harrison, R. F. (2003). Online pattern classification with multiple neural network systems: An experimental study, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews* **33**(2): 235–247.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Machine Learning* **2**: 285–318.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm, *Information and Computation* **108**: 212–261.

Liu, Y. and Yao, X. (1999a). Ensemble learning via negative correlation, *Neural Networks* **12**: 1399–1404.

Liu, Y. and Yao, X. (1999b). Simultaneous training of negatively correlated neural networks in an ensemble, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* **29**(6): 716–725.

Liu, Y., Yao, X. and Higuchi, T. (1999). Evolutionary ensembles with negative correlation learning, *IEEE Transactions on Evolutionary Computation* **4**(4): 380–387.

Marshall, J., Brown, G. and Kovacs, T. (2007). Bayesian estimation of rule accuracy in ucs, *Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation*, pp. 2831–2834.

Martin, B. (1995). *Instance-based learning: Nearest neighbor with generalization*, Master's thesis, University of Waikato, Hamilton, New Zealand.

Mauchly, J. W. (1940). Significance test for sphericity of a normal n-variate distribution, *Annals of Mathematical Statistics* **11**: 204–209.

Minku, F. L., Inoue, H. and Yao, X. (2009). Negative correlation in incremental learning, *Natural Computing Journal - Special Issue on Nature-inspired Learning and Adaptive Systems* **8**(2): 289–320.

Minku, F. L. and Yao, X. (2008). On-line bagging negative correlation learning, *Proceedings of the International Joint Conference on Neural Networks (IJCNN08), Part III*, Hong Kong, pp. 1375–1382.

Minku, F. L. and Yao, X. (2009). Using diversity to handle concept drift in on-line learning, *Proceedings of the International Joint Conference on Neural Networks (IJCNN09)*, Atlanta, pp. 2125–2132.

Minku, L. L., White, A. and Yao, X. (2010). The impact of diversity on on-line ensemble learning in the presence of concept drift, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* **22**: 730–742.
URL: *http://dx.doi.org/10.1109/TKDE.2009.156*

Minku, L. L. and Yao, X. (2010). DDD: A new ensemble approach for dealing with concept drift, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* p. 16p. (accepted).

Mitchell, T., Carbonell, J. G. and Michalski, R. S. (1988). *Machine Learning: a guide to current research*, Kluwer Academic Publishers, USA.

Molina, L., Belanche, L. and Nebot, A. (2002). Feature selection algorithms: A survey and experimental evaluation, *IEEE International Conference on Data Mining (ICDM'2002)*, pp. 306–313.

Montgomery, D. C. (2004). *Design and Analysis of Experiments*, 6th edn, John Wiley and Sons, Great Britain.

Narasimhamurthy, A. and Kuncheva, L. I. (2007). A framework for generating data to simulate changing environments, *Proceedings of the Twenty Fifth IASTED International Multi-Conference on Artificial Intelligence and Applications (AIA'07)*, Innsbruck, Austria, pp. 384–389.

Neurotech (2009). PAKDD 2009 data mining competition. Neurotech Ltd.
URL: *http://sede.neurotech.com.br:443/PAKDD2009/*

Newman, D. J., Hettich, S., Blake, C. L. and Merz, C. J. (2010). UCI machine learning repository. University of California, Irvine, Dept. of Information and Computer Sciences.
URL: *http://archive.ics.uci.edu/ml*

Nishida, K. (2008). *Learning and Detecting Concept Drift*, PhD thesis, Hokkaido University, Japan.
URL: *http://lis2.huie.hokudai.ac.jp/∼knishida/paper/nishida2008-dissertation.pdf*

Nishida, K. and Yamauchi, K. (2007a). Adaptive classifiers-ensemble system for tracking concept drift, *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics (ICMLC'07)*, Honk Kong, pp. 3607–3612.

Nishida, K. and Yamauchi, K. (2007b). Detecting concept drift using statistical testing, *Proceedings of the Tenth International Conference on Discovery Science (DS'07) - Lecture Notes in Artificial Intelligence*, Vol. 3316, Sendai, Japan, pp. 264–269.

Oza, N. C. and Russell, S. (2001a). Experimental comparisons of online and batch versions of bagging and boosting, *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'01)*, ACM Press, New York, pp. 359–364.

Oza, N. C. and Russell, S. (2001b). Online bagging and boosting, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS'01)*, Morgan Kaufmann, Key West, USA, p. 105112.

Oza, N. C. and Russell, S. (2005). Online bagging and boosting, *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, Institute for Electrical and Electronics Engineers, New Jersey, pp. 2340– 2345.

Pierce, C. A., Block, R. A. and Aguinis, H. (2004). Cautionary note on reporting eta-squared values from multifactor anova designs, *Educational and Psychological Measurement* **64**: 916–924.

Polikar, R., Udpa, L., Udpa, S. S. and Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews* **31**(4): 497–508.

Prechelt, L. (1994). PROBEN1 - a set of neural network benchmark problems and benchmarking rules, *Technical Report 21/94*, Fakultt fr Informatik, Universitt Karlsruhe, Karlsruhe, Germany.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco.

Ramamurthy, S. and Bhatnagar, R. (2007). Tracking recurrent concept drift in streaming data using ensemble classifiers, *Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA'07)*, Cincinnati, Ohio, pp. 404–409.

Raymond, E. S., Relson, D., Andree, M. and Louis, G. (2007). Bogofilter, v.1.1.5.
URL: *http://bogofilter.sourceforge.net*

Rossi, C., Barrientos, A. and Del Cerro, J. (2007). Two adaptive mutation operators for optima tracking in dynamic optimization problems with evolution strategies, *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, ACM, London, pp. 697–704.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition* **I**: 318–362.

Schaal, A. and Atkeson, C. (1998). Constructive incremental learning from only local information, *Neural Computation* **10**: 2047–2084.

Schapire, R. (1990). Strength of weak learning, *Machine Learning* **5**: 197–227.

Schlimmer, J. C. and Fisher, D. (1986). A case study of incremental concept induction, *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, USA, pp. 496–501.

Schlimmer, J. C. and Granger Jr., R. H. (1986). Incremental learning from noisy data, **1**: 317–354.

Schlimmer, J. and Granger, R. (1986). Beyond incremental processing: Tracking concept drift, *Proceedings of the Fifth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, pp. 502–507.

Scholz, M. and Klinkenberg, R. (2005). An ensemble classifier for drifting concepts, *Proceedings of the Second International Workshop on Knowledge Discovery from Data Streams(IWKDDS'05)*, Porto, Portugal, pp. 53–64.

Scholz, M. and Klinkenberg, R. (2007a). Boosting classifiers for drifting concepts, *IDA - Special Issue on Knowledge Discovery from Data Streams* **11**(1): 3–28.

Scholz, M. and Klinkenberg, R. (2007b). Boosting classifiers for drifting concepts, *Intelligent Data Analysis (IDA) - Special Issue on Knowledge Discovery From Data Streams* **11**(1): 3–28.

Seipone, T. and Bullinaria, J. (2005). Evolving improved incremental learning schemes for neural network systems, *Proceedings of the 2005 IEEE Congress on Evolutionary Computing (CEC'2005)*, Piscataway, NJ, pp. 273–280.

Stanley, K. O. (2003). Learning concept drift with a committee of decision trees, *Technical Report UT-AI-TR-03-302*, Department of Computer Sciences, University of Texas at Austin, Austin, USA.

Street, W. and Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification, *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (KDD'01)*, ACM Press, New York, pp. 377–382.

Tang, E. K., Sunganthan, P. N. and Yao, X. (2006). An analysis of diversity measures, *Machine Learning* **65**: 247–271.

Tang, K., Lin, M., Minku, F. L. and Yao, X. (2009). Selective negative correlation learning approach to incremental learning, *Neurocomputing* **72**: 27962805.

Tsymbal, A. (2004). The problem of concept drift: Definitions and related work, *Technical Report TCD-CS-2004-15*, Trinity College Dublin, Ireland.

Tsymbal, A., Pechenizkiy, M., Cunningham, P. and Puuronen, S. (2006). Handling local concept drift with dynamic integration of classifiers: Domain of antibiotic resistance in nosocomial infections, *Proc. of the 19th IEEE International Symposium on Computer-Based Medical Systems (CBMS'06)*, Salt Lake City, UT, pp. 56–68.

Ueda, N. and Nakano, R. (1996). Generalization error of ensemble estimators, *Proc. of International Conference on Neural Networks*, Morgan Kaufmann Publishers, San Francisco, CA, p. 9095.

Utgoff, P., Berkman, N. and Clouse, J. (1997). Decision tree induction based on efficient tree restructuring, *Machine Learning* **29**(1): 5–44.

Valiant, L. (1984). A theory of the learnable, *Communications of the ACM* **27**(11): 1134–1142.

Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Applications* **16**(2): 264–280.

Wang, H., Fan, W., Yu, P. S. and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers, *Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining (KDD'03)*, ACM Press, New York, pp. 226–235.

Wang, Z., Yao, X. and Xu, Y. (2004). An improved constructive neural network ensemble approach to medical diagnoses, *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'04), Lecture Notes in Computer Science*, Vol. 3177, Springer, Exeter, UK, pp. 572–577.

Wen, W. X., Jennings, A. and Liu, H. (1992). Learning a neural tree, *Proceedings of the 1992 International Joint Conference on Neural Networks (IJCNN'92)*, Vol. 2, Beijing, China, pp. 751–756.

Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden context, *Machine Learning* **23**: 69–101.

Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning, *Neural Networks* **16**(10): 1429–1451.

Witten, I. H. and Frank, E. (2000). *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, San Francisco.

Yaochu, J. and Branke, J. (2005). Evolutionary optimization in uncertain environments - a survey, *IEEE Transactions on Evolutionary Computation* **9**: 303–317.

Yates, F. (1934). Contingency table involving small numbers and the χ^2 test, *Journal of the Royal Statistical Society Supplement* **1**: 217–235.

Yule, G. (1900). On the association of attributes in statistics, *Phil. Trans.* **A, 194**: 257–319.